Using Artificial Neural Networks to Model Running Speed in Orienteering

Øystein Jaren Samuelsen

Master of Science in Computer Science

Submission date:	July 2012
Supervisor:	Helge Langseth, IDI
Co-supervisor:	Boye Annfelt Høverstad, IDI
	Jan Kocbach, Christian Michelsen Research

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

This thesis concerns the modeling of running speed in orienteering by means of multi-layered feed forward artificial neural networks with Backpropagation learning, using GPS tracks of orienteers, an orienteering map and a digital elevation model as the basis for the training data. A learning system was implemented and tested with GPS data collected by a test subject. A trained speed model was applied in a third-party application for arithmetic analysis of route choices. The proposed method is shown to have potential, even though the results as of now are not good enough to be considered useful to orienteers.

Acknowledgements

I would like to thank my supervisors Boye Annfelt Høverstad, Jan Kocbach and Helge Langseth for their invaluable advice during this thesis. I also thank Ann Karin Halvorsen of Statens Vegvesen, Region Sør, for helping me process the laser-based digital elevation models. Finally, I must thank Felix Arnet for letting me use his software for the purposes of this thesis, and for his invaluable help with getting the software to work in concert with my own implementation, something which included many hours of work on his part. Thank you very much.

1 Contents

1 Introduction

This study was performed to fulfill the requirements of a M.Sc thesis in Computer Science at the Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, Norway. It concerns the analysis of GPS tracks of orienteering athletes to model the relationship between various terrain features and expected running speed. This relationship is expected to be quite complex, with dependencies between the various factors that affect the speed. Finding a reliable running speed model can allow for accurate route choice analysis and calculation of optimal routes using digital maps and elevation models. This may prove useful for athletes and coaches during preparation for important events and analysis of orienteering technique, as well as for course planners in optimal positioning of the controls of an orienteering course. Uses in visualization for TV production purposes are also imaginable.

1.1 Background and motivation

1.1.1 Orienteering

Orienteering is a sport in which the athletes – called *orienteers* or simply *runners* – are required to navigate at high speed through unfamiliar terrain using a map and compass. An orienteering course consists of a number of control points, typically called *controls*, which must be visited in the specified order as indicated on the map. Each control is marked on the map by a small, red circle around the control point, and the order is indicated by numbering, as well as straight line segments connecting the controls. Such a segment from one control to the next is called a *leg*. In the terrain, the control is marked by a red and white flag. For each leg, the athlete has to use the map to make a decision about which route to take, and then navigate through the terrain to the next control. Figure 1-1 shows an example of an orienteering map with a course on it.

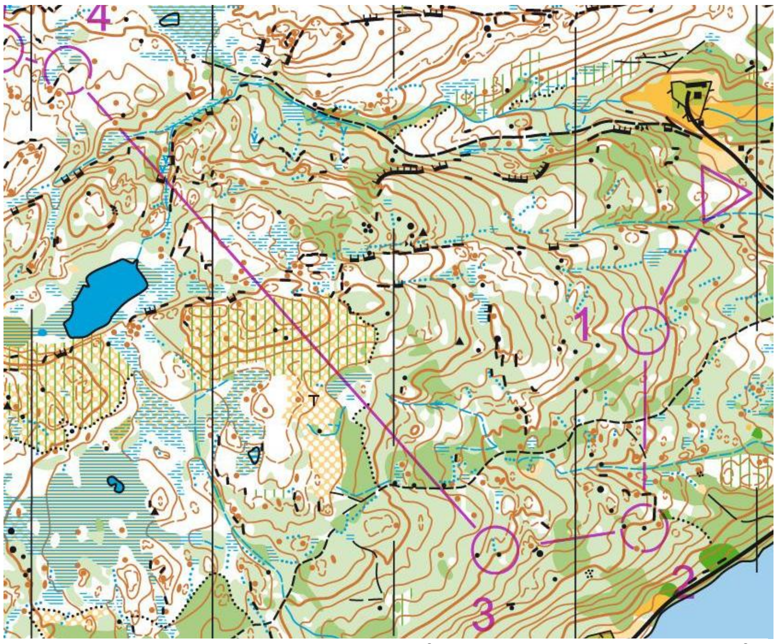


Figure 1-1: Orienteering map. Example of an orienteering map with part of an orienteering course. Excerpt from **(Forbord & Bjartnes, 2010)**

1.1.2 The route choice problem

The route choice problem – choosing the best possible route to the next control – is obviously a central part of orienteering. A bad route choice can result in time loss of anything from a few seconds to several minutes. However, taking too much time to decide on a route may be just as bad as making a sub-optimal choice. Thus, orienteers must be highly adept at spotting the most promising route options for any given leg, and they must preferably do this before they get to that particular leg, i.e. while running a previous leg. Figure 1-2 shows an example of a route choice problem and the routes that were selected by three orienteers.

Elite orienteers spend considerable amounts of time studying orienteering maps and route choice problems, analyzing their previous competitions and training sessions and preparing for future competitions. Specific preparations in advance of an important event is especially crucial if it is to be held in a type of terrain that is unfamiliar to the athlete, i.e. the characteristics of the terrain are very different from what she is used to. For instance, making route choices in a flat coastal terrain is very different from making route choices in a steep mountain terrain.

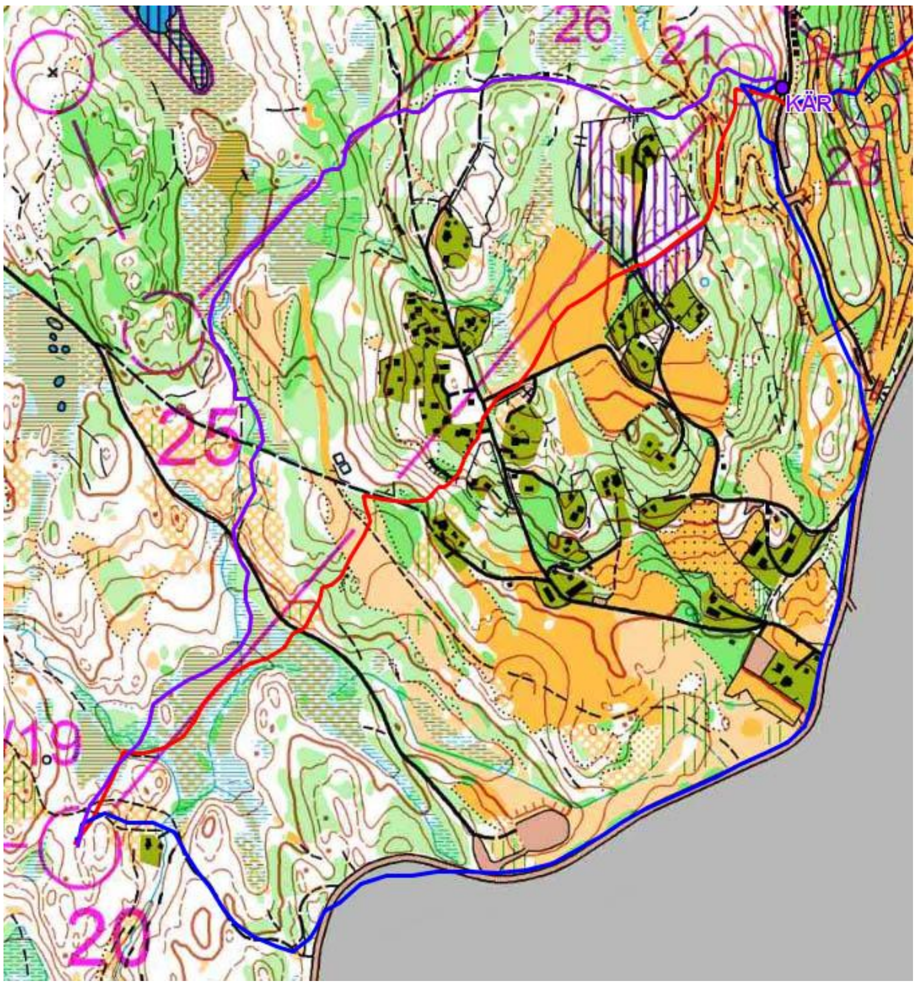


Figure 1-2: Route choice leg. Example of a typical route choice problem. There are many ways to get from control 20 to control 21. The GPS tracks of three orienteers are shown. Excerpt from **(Forbord, et al., 2010)**

The route choice is made so as to avoid unnecessary ascent/descent, dense vegetation and undesirable running surfaces (rocky ground, marshes etc.), all of which can hinder the running speed. The avoidance of these hindrances must be balanced against the added distance that must be covered in order to avoid them. Finding this balance is a difficult task, and is not the same in different types of terrain. The map can only tell you so much; it is also necessary to be familiar with the terrain type. For instance in some types of forest the ground is known to be generally rough and uneven, and this suggests that it might pay off to choose a longer route on paths/roads (if such an option exists) than to go straight through the forest. In other types of forest the ground is known to be smooth and highly runnable, which suggests that it is usually best to go as straight as possible, unless of course there are other factors (such as elevation and vegetation) that suggest a longer route is faster.

1.1.3 Speed Model for Orienteering

To be able to select the best possible route for any given leg, the orienteer must have some model in his head of how the different terrain features affect the running speed. The use of GPS-enabled sports watches and sophisticated software for after the fact analysis orienteering trainings and races can help orienteers identify their mistakes and see where they chose sub-optimal routes. However, to the author's knowledge, there is currently no tool which can automatically harness the vast amounts of information implicitly available in the map and GPS data and use it to build a complete model of the relationship between terrain features and running speed. That is the goal of this thesis.

1.1.4 Arithmetic route choice analysis

The orienteering map details terrain features such as vegetation density, running surface and of course elevation (contours). These all affect the speed with which the athlete is able to run. If the relationship between these features and running speed is completely known, one can theoretically calculate the optimal route for any given leg, as well as the time required to run any particular route choice for the leg. Arnet (2009) showed that such route choice analysis is possible. However, the speed model employed, i.e. the part of the system which dictates the actual running speed for any given direction on any point on the map, was a very general one, which did not take differences between terrain types into account. As discussed in Section 1.1.2, different terrain types call for different route choice considerations.

This thesis looks into the application artificial neural network (ANN) learning to the problem of devising running speed models for orienteering. The training data consists of GPS tracks of athletes running in the forest, combined with a digital orienteering map and a digital elevation model (DEM). The elevation model can be derived from the orienteering map's contours or obtained from laser scans of the terrain. From this combination one can get a large number of training cases that map the set of terrain features at the current location to the current running speed. Machine learning methods such as ANN learning can be used to learn this mapping, i.e. the speed model. The speed model output by such a method, if sufficiently accurate, could be used to improve arithmetic route choice analysis such as that described in (Arnet, 2009).

The application of ANN learning seems a reasonable approach, as the space of possible terrain feature combinations is very large. Though the number of map feature combinations alone is not overwhelming, the addition of running slope (rate of ascent/descent in running direction) and terrain slope (steepness of terrain, altitude gradient magnitude) which are of course continuous values, makes for an unbounded number of different terrain feature combinations. Furthermore, one cannot necessarily assume independence between the various factors that hinder the running speed. For instance, a steep negative running slope typically hinders running speed when compared to flat running, because of the unevenness and vegetation of the forest ground. (Attempting to run at full speed down a steep hill in the terrain would likely result in injury.) However, with the presence of a path or a road, or even an open grass field, things are quite different. The smooth running surface allows for high speed even with a large negative running slope. Such dependencies may be captured by an artificial neural network.

Furthermore, it is likely that the various speed factors will be different for different types of terrain, as suggested in the previous section. A machine learning approach will allow for the creation of speed models for specific types of terrain (and specific runners). One must simply input training data from competitions in similar terrain (and of the specific runner) to obtain the appropriate model.

1.1.5 Possible applications

A system that can learn the terrain-speed relationship from GPS data would be a highly versatile tool for orienteers and their coaches when preparing for important competitions. By training and collecting GPS tracks in relevant terrain (terrain similar to that in which the competition is to be held), the orienteer would be able to simply input the GPS data (as well as map and DEM) to the system, which would then construct a model of the relationship specific to the type of terrain and the particular orienteer. This model could then be used as the basis for automated calculation of optimal routes, the degree of "sub-optimality" of other routes etc., using software such as that described by (Arnet, 2009). When preparing for competitions in unfamiliar terrain types, a program that can reliably show you the best route choice and the amount of time you lose by choosing differently, i.e. a *route choice model*, would be an invaluable tool.

One can imagine using the speed model learning system to create a speed model based on GPS data from a group of elite orienteers. The resulting model should represent an average of the participating runners, i.e. a speed model for a "typical" elite orienteer. A personal speed model would be more useful, however, since it is an orienteer's own strengths and weaknesses, not those of other runners, that matter when a route choice is to be made. For instance, some runners are better than others at running in dense forest, while others again may be particularly fast on paths and roads. Still, comparing the personal speed model with a more general model (this one based on data from several elite runners) may allow the athlete to identify her own strengths and weaknesses as a runner. The two speed models could both be used to compute the time cost of running a particular route, for instance one that goes straight through an area of dense vegetation. If the personal speed model computes a higher cost than the general speed model, it is an indication that the athlete in question is slower than other athletes in dense vegetation. With this information the athlete is better equipped to know what to focus on during training, in this case to run in dense vegetation.

Course planners would likely also benefit from a reliable route choice analysis tool. An orienteering course is carefully planned to provide the most interesting orienteering challenges the terrain can provide. A tool that can show you the best route choices as you move the controls around would be highly useful. For instance, it would allow for precise positioning of controls with the purpose of making one route choice slightly better than another, or trying for a certain number of route choices that are almost equally good, resulting in a difficult decision for the athletes.

Taking this one step further, one can even imagine automatic course setting. A good orienteering course should preferably contain as many interesting route choice challenges as possible within the constraints of the course and map (total length, area restrictions etc). Thus an important criterion for the quality of a leg is how interesting it is from a route choice perspective. One could define a function to quantify this criterion and use it as the fitness function of an evolutionary algorithm. The fitness function could for instance include the number of distinct routes within a certain range of time loss compared to the optimal route (distinct meaning there must be a certain distance between the various routes). One could also possibly evolve a complete orienteering course by seeking to maximize the number of interesting route choice problems within the constraints of the course and map. The output of such a process would likely be a valuable asset as a basis for further development of the course; a sort of "automated brainstorming".

Finally, visualizations of calculated route choices could be useful in TV production of orienteering, as it may allow viewers that are not proficient in map reading to understand the subtleties involved in making the right route choice.

1.2 Research Questions

In order to concretize the problems that will be addressed in the thesis, this section lists a set of research questions. The overall goal is to find out whether artificial neural network learning can be successfully employed in devising speed models for orienteering, in particular to be used for arithmetic route choice analysis. The focus will be on personal speed models, i.e. speed model based on the GPS data from a single orienteer. This leads to the following research questions:

1. ***Is it possible to find useful running speed models for orienteering by means of artificial neural network learning with training cases consisting of GPS tracks from an orienteer combined with digital orienteering maps and digital elevation models?*** *How accurate is the resulting model when applied to a test set?*

2. ***Is the resulting speed model accurate enough to be useful for arithmetic route choice analysis?*** *Is the time calculated for a particular route through the forest close to the time it takes to run that route in practice? Does the best route choice identified by the model for a given leg match the best route choice in practice?*

1.3 Thesis Structure

Chapter 2 deals with the theory and background that is necessary in order to understand and appreciate the rest of the thesis. Chapter 3 describes the proposed method and the implemented system. Chapter 4 presents the experiments that were conducted to test the method and system, as well as the results of these experiments. Some closing remarks and future work is given in Chapter 5.

2 Theory and Background

This chapter gives a brief overview of the theory and background that is necessary to appreciate the rest of the thesis, as well as an overview of previous work in the areas of orienteering route choice analysis and prediction of running speed.

2.1 Orienteering Route Choice Theory

Weltzien (1983) states that though the individual orienteer possesses considerable knowledge on the topic of route choice from many years of experience, knowledge on the topic in printed form is sparse. Unfortunately this statement still holds true today. Although route choice is a frequently discussed topic at seminars and training camps where elite orienteers and coaches participate, there is relatively little actual research available. To the author's knowledge, the only attempt at a comprehensive treatment of the topic is (Weltzien, 1979) and (Weltzien, 1983). Through empirical studies of the route choices made by elite orienteers, Weltzien developed a crude model of the relationship between terrain characteristics and running speed in Nordic terrain. Based on this model, a set of guidelines for making the correct route choice were given. Even though it is old, and there are now far better ways to collect empirical data about route choices (using GPS), Weltzien's work highlights some useful points that should be just as valid today as they were at the time of writing. The route choice situation is described as a balancing act between three possibly competing criteria, usually in the following order of priority:

1. Which route is the fastest?
2. Which route is the safest? In other words, which route has the lowest risk of time loss from orienteering mistakes?
3. Which route is the most energy conserving?

The third criterion is ignored in this thesis, as we are primarily concerned with finding the fastest route choice on a single leg at a time. In long competitions, this criterion may need to be considered. The second criterion is difficult to handle, as the combination of terrain features at the current point alone is not enough to say anything particularly useful about the risk of making mistakes. Besides, for an elite orienteer, in most types of terrain, orienteering mistakes are few and small. Therefore the primary concern will always be to find the fastest route.

(Myrvold, 1996) investigated the certainty with which one can say that a particular route is in fact the best choice. The conclusion of this work was that for a real route choice problem, that is a route choice leg which elite orienteers consider interesting, it is impossible to find a statistically significant difference in time between (at least the best two) alternative routes just by looking at the time taken by various runners on the different route alternatives. If a statistically significant time difference can be found, which means that there is *one* route choice that is best for all runners, then the best route is so obvious (to elite orienteers) that there is essentially no route choice to make. Rather, the best route is dependent on the individual orienteer’s skill. Myrvold found support for this claim in previous route choice studies, especially (Weltzien, 1979), where the participating orienteers ran two or three different predetermined route choices for each leg. The runners were encouraged to begin with the route that they considered to be the best alternative. In the vast majority of cases the runners chose correctly the first time, i.e. they achieved their best time for the leg with the route they regarded as the best.

Weltzien took this as a sign that the runners get tired as time goes on, but Myrvold points out that this could also indicate that the runners know their own strengths and weaknesses well. This is very important knowledge for any orienteer in order to choose the best route for that particular orienteer. There may still be a best route for the average elite orienteer, but Myrvold claims it is not possible to prove with any significance that any particular route is in fact that best route. In any case, the problem of finding the best route for the average orienteer is somewhat less interesting than the problem of finding the best route for a particular orienteer, since that information is far more useful to the orienteer in question.

2.2 Global Navigation Satellite Systems

The *NAVSTAR Global Positioning System* (*GPS*) is one of the two currently fully operational *global navigation satellite systems* (GNSS), the other being the Russian *Global Navigation Satellite System (GLONASS)*. Other systems, such as the European *Galileo* and the Chinese *Compass,* are currently under development. As of today, GPS is by far the most well known and most used GNSS.

A GNSS is a system of satellites which provides accurate positional information to compatible receivers all over the world. The satellites broadcast radio signals containing information about the satellite's current position and the current time (using atomic clocks). A compatible receiver on the ground or in the air can use this information to compute the distance to each satellite and thus the receiver's position in 3D (ground coordinates plus altitude), as long as at least four satellites are within sight. More available satellites improves the accuracy. Under good conditions, consumer grade GPS receivers can pinpoint their ground position (2D) with an accuracy down to about three meters, while altitude accuracy (3D) typically is quite poor (especially if most of the satellites are located near the horizon). The accuracy of the GPS coordinates (in both 2D and 3D) can be negatively affected by factors such as atmospheric conditions, radio signal noise, satellite positions and obstruction of satellites by large buildings, terrain forms or dense foliage.

For orienteering purposes, inaccuracies caused by obstructing terrain forms and dense foliage may cause problems in some cases, but if the GPS receiver has been allowed to get a good satellite fix in an open location before the start of the race (with as many satellites as possible in sight), it will typically be able to relocate quickly after a satellite has been temporarily out of sight. GPS altitude accuracy is not of any concern in this thesis, as a digital elevation model is used as the source for altitude data. Only the 2D coordinates from the GPS receiver are used to find the correct location on the ground.

2.2.1 GPS and Orienteering

With the proliferation of relatively cheap GPS receivers in the past few years, some aspects of the orienteering sport have gone through a revolution. Traditionally orienteering has been seen as a somewhat obscure sport, mostly because it has been impossible - or at best very difficult - for spectators to follow the progress of a race, either from the arena or on TV. The runners could typically only be seen at a few well chosen TV spots, in between which spectators had to make do with split times from radio controls. The use of GPS receivers with mobile communication capabilities (worn on the runner's back) has dramatically improved the spectator-friendliness of orienteering. Using a GPS tracking system such as GPSSeuranta (Varis, 2012), it is now possible to follow the progress of the athletes in real time, in the form of moving dots superimposed on the orienteering map. Mass start simulations from arbitrary controls can also be done, making for interesting visualizations that reveal the differences between route choices as well as the mistakes made by specific runners. A screenshot from GPSSeuranta is shown in Figure 2-1.

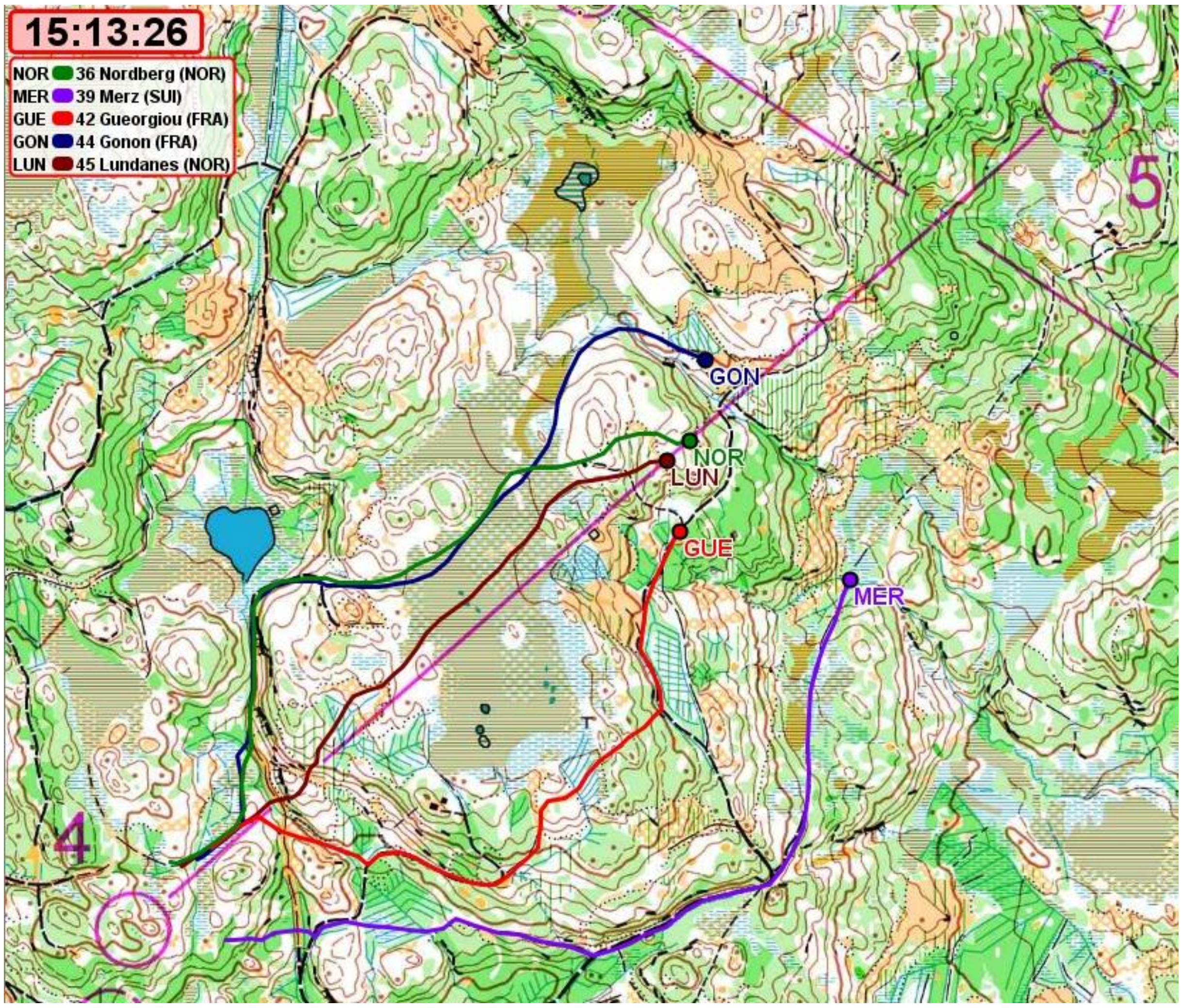


Figure 2-1: GPSSeuranta. A screenshot from the live tracking system GPSSeuranta (Varis, 2012). The image shows five orienteers in a simulated mass start from the fourth control of the WOC2010 Long Distance Final. A named dot indicates an orienteer's current position, while the "tail" of a dot shows the orienteer's route during the past few minutes (in this case the past five minutes).

GPS receivers have also made their way into personal sports watches and small devices known as *GPS loggers*. These are used in large scale by orienteers to continually log positions during training, and then for subsequent analysis on a computer using various software tools. An overview of such software is listed in (Kocbach, 2011). The possibilities brought on by these tools have revolutionized the way orienteers do after the fact analysis of trainings and competitions. Every little mistake and slight hesitation is revealed. This allows an orienteer to identify his trouble spots and prioritize future training accordingly.

2.3 Artificial Neural Network Learning

2.3.1 Overview of Artificial Neural Networks

An *artificial neural network* (ANN) is a type of function approximator. It is an interconnected network of simple processing units, typically called nodes or neurons, which maps a set of real-valued inputs - the *input vector* - onto a set of real-valued outputs - the *output vector*. The connections between neurons are directed and weighted, such that the input to each neuron is the sum of weighted outputs from its *pre-synaptic* neurons. The weighted sum is transformed by an *activation function*. A common choice of activation function is the logistic function, also called the *Sigmoid* function

$$\sigma(y) = \frac{1}{1 + e^{-\alpha y}} \tag{2.1}$$

where y is the weighted sum of inputs to the neuron and $\alpha > 0$ is a constant. The result of the activation transformation is output on the connections to the *post-synaptic* neurons. The terms pre-synaptic and post-synaptic refer to the primary inspiration for ANNs - a biological brain - where a synapse denotes the connection between two neurons that allows them to communicate.

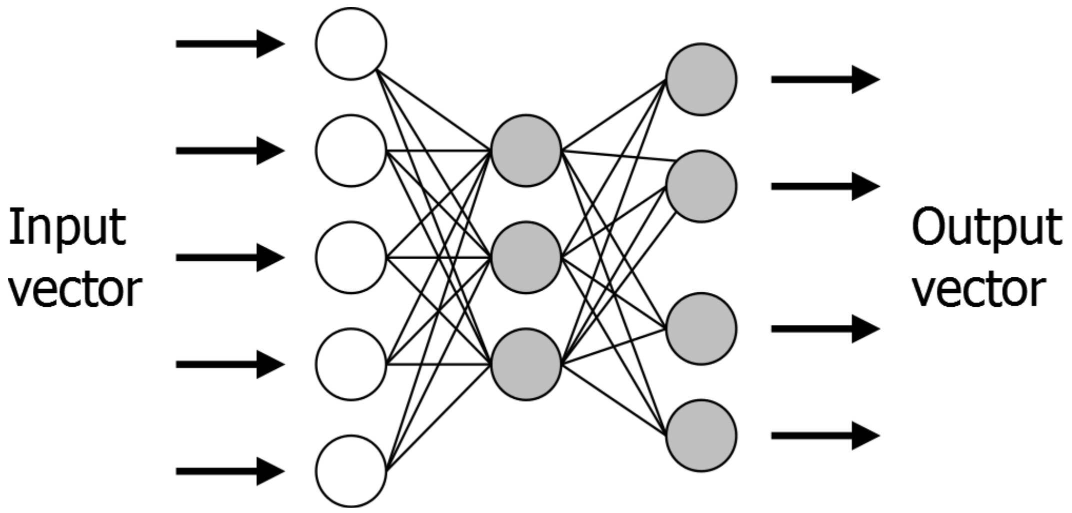


Figure 2-2: Schematic view of a layered feed forward ANN. Note that this network contains two layers - one hidden layer and the output layer. The input layer (white) is not actually a layer, as its only job is to map the input vector onto the nodes of the first hidden layer.

The most commonly used type of ANN is a *layered feed forward network*. An example of such a network can be seen in Figure 2-2. A feed forward network contains no cycles, i.e. there is no way for the output of any particular neuron to contribute to the input of that same neuron, either directly or via other nodes. This also implies that there is no notion of time or temporal sequences. A feed forward network simply returns an output for a particular input. In a layered feed forward network, the nodes are arranged in layers, such that the outputs of nodes in layer l_i only map to inputs of nodes in layer l_{i+1}. The last layer in the network is the *output layer*, which means that the values output from the nodes in that layer make up the output vector of the network. In the simplest kind of layered feed forward networks the output layer is the only layer, and the input vector maps directly to the output layer. Such a network has limited usefulness, however, as it can only represent linearly separable functions. By introducing *hidden layers* of nodes with non-linear activation functions between the input vector and the output layer, more complex target functions can be approximated. In particular, (Cybenko, 1989) established that an ANN with a single hidden layer using a sigmoidal activation function and output layer using a linear activation function can represent any continuous function, provided the number of hidden neurons is sufficient. The reason why the Sigmoid function is not used in the output layer is that its output range is limited to $(0, 1)$.This can be somewhat circumvented by multiplying the target outputs by a scaling factor prior to training in order to scale them into the $(0, 1)$ range. Later, when the network is queried with unseen inputs, the resulting outputs are scaled using the inverse scaling factor to obtain the real output values. This approach may cause problems if many target values are near the limits of the output range, i.e. near 0 or 1, since the Sigmoid function only approaches 1 (0) as the input approaches infinity (negative infinity).

It is necessary to use a non-linear activation function (such as the Sigmoid function) in the hidden layer(s). Using a linear activation function here would be pointless, because a linear combination of linear functions is still just a linear function, and so adding the hidden layer would not increase the representational power of the ANN.

Knowing that a single hidden layer network can approximate any continuous function, the problem remains to actually devise a network which approximates the function in question. This mainly involves two steps:

1. Devising a network topology, i.e. deciding how to encode the input data and output data as vectors of real values, as well as deciding on the number of hidden layers and the number of neurons in each layer, and what activation function to use. (Even though one hidden layer is enough in theory, more may be useful in practice. For instance, (Ciresan, et al., 2010) used an ANN consisting of 5 hidden layers with a total of 7500 neurons for handwritten digit recognition with very convincing results.)

2. Tuning the connection weights until the network approximates the target function with sufficient accuracy.

The problem of selecting the appropriate number of layers and neurons can to some extent be automated as well (see for instance (Fahlman & Lebiere, 1990)). But trial and error is still a prevalent approach in this regard.

Another important issue to consider is whether the input vector needs to be normalized, and what type of normalization method to use. This means that the different inputs should be constrained to the same range. When the activation function used is the Sigmoid function, this range should be $(0,1)$. If the ranges of the different inputs are vastly different, the neurons in the first hidden layer will not be able to differentiate between the input patterns. This is demonstrated in (Kim, 1999). A number of different normalization methods were studied by (Kim, 1999) and (Jayalakshmi & Santhakumaran, 2011).

2.3.2 Training an Artificial Neural Network using Backpropagation

Once the network topology is established, a multi-layered feed forward ANN can be trained by means of the Backpropagation algorithm, which is described by (Rumelhart, et al., 1986). It is a supervised learning algorithm. In other words, a number of training examples mapping input vectors to the correct output vectors must be available.

The algorithm is a generalization of the gradient descent learning rule for single layer ANNs. It works by first initializing the ANN's weights to random values within a specified range. A typical choice is $(-0.5, 0.5)$. Then the training examples are presented to the ANN over several epochs. In each epoch, all the training examples are presented in random order. For each training example the ANN computes the output vector, which is compared with the desired output vector as given by the training example, and the difference (the error) is computed. When all the training examples have been presented to the ANN, i.e. at the end of the epoch, a total error measure taking into account the error of each output summed over all training examples is computed.

The error is then back-propagated through the network, and each weight of the ANN is updated by an amount proportional to the derivative of the total error with respect to the weight, in the opposite direction of said derivative. In simple terms, each weight throughout the network can be attributed part of the responsibility for the total error at the output layer, and the backpropagation algorithm essentially assigns this responsibility to each weight and updates the weight accordingly to reduce the total error. The size of the weight update is also regulated by the *learning rate*, which is a number in the range $(0, 1)$, typically in the lower part of the range. Choice of learning rate has an impact on convergence of the ANN. With a too low learning rate the network will learn slowly and may easily get stuck in local minima. With a too high learning rate the network will learn quickly, but may overshoot and miss the global minimum completely. A common strategy is to start with a high learning rate and reduce it as training progresses.

Throughout training, the ANN's performance is periodically tested on a *validation set*, containing training examples from the same distribution as the training set. The usual approach is to divide the whole set of training examples into two sets. One of them is used for training, while the other is used for validation. This strategy called as *cross-validation*. What one typically sees after many iterations of training is that the mean error over the validation set - the *validation error* - stops to decrease and starts increasing, while the mean error over the training set - the *training error* - keeps decreasing. This indicates that the ANN is beginning to *overfit* the training data. This means that the ANN is tuning into patterns in the training data that are not representative of the original distribution. Training should continue until we are confident that the mean validation error has reached its lowest point. The ANN that resulted in the lowest validation error during training is returned as the final ANN.

In order to make the best use of the available training data, as well as reduce the variability in the error estimate (over a test set) and the predictions output by trained networks, a technique known as *k-fold cross-validation* can be used. The whole set of training examples is divided into k sets of equal size. Then k ANNs are trained, each time using one of the k sets as the validation set, while the rest are combined and used as the training set. After training, each of the k sets has been used exactly once for validation. The trained ANNs then work as a *committee* or *ensemble*. Subsequent queries to the committee are processed by each member ANN, and the outputs are combined to produce the final output of the committee. This can be done by simple averaging, weighted averaging, order statistics and possibly other methods. To evaluate the generalization performance of the committee, a separate test set (not used for training or cross-validation) can be applied. It has been shown that committees of ANNs often outperform any single ANN on generalization performance. See for instance (Hashem, 1997).

Even though ANN learning by Backpropagation is not guaranteed to find the globally optimal set of connection weights, it has been shown to be a versatile tool fit for solving a vast variety of tasks. These tasks can roughly be separated into two categories depending on the nature of the ANN's output. The first category involves discrete outputs, i.e. classification of an input case into one of several discrete classes. Examples include handwritten digit recognition (Ciresan, et al., 2010) and face detection (Rowley, et al., 1998). In such applications, the output layer typically contains one output neuron for each class (e.g. all the different digits, face/no face), and the output neuron with the highest activation level is taken as the computed class for the current input. In the other category, the output is one or more continuous values, so the problem is that of regression. An example is prediction of stock prices (Freislaben, 1992). The problem which is the topic of this thesis, prediction of running speed, falls into the second category of applications.

2.3.3 Variations on Backpropagation

Many variations and modifications of the standard Backpropagation algorithm have been suggested, seeking to improve upon various aspects of it, such as the likelihood of convergence to an optimal solution, speed of convergence, robustness against variations in initial weights etc.

A common variant trains the ANN *online* by updating the weights after the presentation of each training example, rather than at the end of each epoch. The AForge.NET framework, which was used for ANN related tasks in the implemented system for this thesis, uses this approach.

Probably the most well-known variant is the introduction of a *momentum* factor (Rumelhart, et al., 1986). Each time a weight is to be updated, a fraction of the previous update of the weight is added. This may allow the Backpropagation algorithm to escape local optima and areas of the error space where the error gradient is zero.

One of the fastest variations on Backpropagation is known as Resilient Backpropagation, or RPROP (Riedmiller & Braun, 1993). Instead of updating each weight in proportion with the partial derivative of the error, RPROP multiplies the previous weight update by a factor η^- or η^+ ($0 < \eta^- < 1 < \eta^+$), depending on the sign of the partial derivative. If the partial derivative with respect to the weight changed its sign since the last iteration, the previous weight update is multiplied by η^-. If the sign is unchanged, the weight update is multiplied by η^+. This allows the speed of weight change to increase rapidly while the sign of the error derivative is constant, and slow down when the sign fluctuates. Instead of leaving η^- and η^+ as tunable parameters, they were set to the constant values of 0.5 and 1.2 by the algorithm's inventors, as these values seemed to work well independent of the examined problem. An implementation of RPROP is available in the AForge.NET framework, discussed in Chapter 3.

2.3.4 Other Neural Network Approaches

Although layered feed forward networks are the most common type of ANN in use, other alternatives exists. The decision to use a feed forward network implies a stateless assumption. In the case of this thesis: The running speed depends only on the terrain features that can be observed at the current coordinates and time, and not on the speed of previous coordinates. For instance, one could image that a runner gets tired from climbing a steep hill, and that this affects the running speed for some time afterwards. Some type of recurrent network could be used to model such dependencies. The training data would then consist not of individual training examples of terrain features and resulting speed, but of a sequence of such examples. This would require a much larger amount of training data than what was available for this thesis, which would in turn make the method less applicable as a tool for orienteers. Furthermore, since no one has attempted to solve this problem before, it seems reasonable to try a simpler approach, such as feed forward ANN and Backpropagation, as a first step. For these reasons the possibility of using recurrent networks was not considered further for this thesis.

Radial Basis Function Networks (RBFNs) may be another viable approach to the problem. It has been proved that RBFNs, like feed forward ANNs with one hidden layer with non-linear activation function, are universal approximators (Park & Sandberg, 1991). The potential use of RBFNs was however not investigated further in this thesis.

2.4 Related Work

To the author's knowledge, no one has attempted to model the relationship between terrain characteristics and running speed using machine learning techniques, although mathematical models for approximation of the running speed do exist. Section 2.4.1 gives an overview of some of these models, while Section 2.4.2 describes how these models and other techniques have been applied for the purpose of route choice analysis.

2.4.1 Models of Running Speed Based on Terrain Characteristics

Predicting the running speed based on the information given by the map is core to any attempt at making a route choice. Although an orienteer does not compute speeds in his head while planning for a particular leg, the route choice comes down to the orienteers experience of the relative speeds in different kinds of terrain configurations. It is however difficult to verify that the orienteer's beliefs of these relative speeds are in fact true. If an appropriate mathematical or statistical model of running speed in varying terrain could be found, it would be of great help in the task of validating the orienteer's own beliefs of the terrain's influence on his running speed.

Naismith's rule, originating from (Naismith, 1892) has long been a commonly applied heuristic for estimating the time required for walking or running various routes in mountainous terrain. The rule states that the time required to walk a certain route with length Δd and ascent Δh corresponds to the time required to walk a distance in flat terrain equal to $\Delta d + a \cdot \Delta h$, where Naismith's number $a = 7.92$. So 1 m of ascent corresponds to 7.92 m of horizontal travel in terms of the time required. Note that Δh is the accumulated *positive* ascent over the distance Δd. Descent is not considered. Others have attempted to extend Naismith's rule to account for descent. See for instance (Hayes & Norman, 1984). Scarf (2007) found support for Naismith's rule in data from fell running records, with a recommended value of Naismith's number of $a = 8$ for men and $a = 10$ for women. In orienteering, $a = 10$ is typically used, as it simplifies calculations. This makes the rule usable as a heuristic in route choice situations.

When looking at route choice analysis from an arithmetical point of view, i.e. as a computational effort, a complete speed model is needed. In other words, we need to be able to determine the running speed at any point in the terrain, in any direction, i.e. for any combination of terrain features, and not just the rate of climb. Most importantly, one must consider vegetation and running surface. Information about vegetation given on the map is limited to the level of density, ranging from open land to impassable thicket. The type of vegetation (kinds of trees and bushes) can also influence the speed, but this information is not given on the map. The orienteer needs to rely on experience and knowledge of the terrain type in order to take that into account. A similar argument holds for terrain surface. While the map tells the orienteer where there are paths and roads, regular forest surface and rocky ground, there is no information inherent in the map about the actual structure of these surfaces. And while roads and large paths can be expected to be equally runnable in different terrain types, the runnability of regular forest surface and possibly smaller paths can be very different in different types of terrain. This means that a speed model that works well in one type of terrain may not work well at all in a different type of terrain. For these reasons, The International Specification for Orienteering Maps (Persson, et al., 2000) specifies only a very approximate range of speed factors for the various degrees of dense vegetation, in relation to "normal speed", which is presumably the speed in regular forest, i.e. the white parts of the orienteering map. There are three degrees of dense vegetation, and these are specified to 60-80 %, 20-60% and 0-20 % of normal speed, respectively. Within a given terrain type, or a limited area, such as the area used for a competition, there will probably not be variations of this magnitude.

(Arnet, 2009) described running speed at a point in the terrain as a function of rate of ascent in running direction, terrain slope and obstruction. Terrain slope in a point is the slope of the terrain in the direction of steepest ascent, i.e. the magnitude of the altitude gradient. This is a crucial factor. For instance, even when the ascent in running direction is zero, a steep terrain slope will have a negative impact on running speed. (Such a situation will occur when running along a hillside, following the same altitude.) Obstruction is a function of vegetation and running surface.

The cost of travelling in a straight line from p_0 to p_1 (where p_0 and p_1 are two points very close to each other) is computed according to the formulas

$$\Delta t \cdot v_0 = \frac{\Delta d}{f_O} + a \cdot \Delta \mathrm{h}_{\mathrm{pos}} + (\mathrm{g}^2 \cdot \mathrm{b}) \cdot \Delta \mathrm{d} \qquad \textbf{(2.2)}$$

$$\Delta t \cdot v_0 = \frac{\Delta d}{f_{road}} + a \cdot \Delta \mathrm{h}_{\mathrm{pos}} \qquad \textbf{(2.3)}$$

where Δt is the time used, v_0 is the speed on a flat road, Δd is the covered distance, f_O is the speed factor due to obstruction O, a is Naismith's number, Δh is the total altitude gain, $\Delta h_{pos} = max(0, \Delta h)$, g is the terrain slope, and b is a slope correction factor, similar in function to Naismith's number. The values of a and b were are set to 7 and 7.5 respectively. The value of the various f_O for vegetation are approximately based the percentage ranges given in (Persson, et al., 2000), which were discussed on the previous page. $f_{road} = 1$ is taken as the speed factor for all paths and roads. Note that the terrain slope is assumed to have no impact on the speed when running on a path/road (roads and paths are normally quite level), which is why there is a separate formula for this case. The computation on the right side of either equation yields the flat road distance that corresponds to the values of Δd, f_O, Δh_{pos} and g, i.e. the distance on a flat road that would take the same amount of time to run as the distance Δd with speed factor f_O, positive altitude gain Δh_{pos} and terrain slope g. Dividing the flat road distance by the real flat road speed v_0 would yield the actual time cost.

2.4.2 Arithmetic route choice analysis

The final speed model described in the previous section was applied by (Arnet, 2009) as the basis for his arithmetic route choice analysis application, hereafter referred to as the *LeastCostPath* application. Because that application will be used to evaluate the speed models produced in this thesis, a detailed description of its workings is given here.

Two models of the terrain are needed to compute speeds and time costs according to the speed model in (2.1): The obstruction model and the DEM. The obstruction model describes horizontal running speed, and is obtained by converting the digital orienteering map into a raster image using a special symbol set. The map symbols that affect horizontal running speed are encoded as various shades of gray, where black is fastest (road) and white is impassable. (This is done using OCAD (OCAD, 2012), which is the de facto standard software for making orienteering maps.) The DEM is an ESRI Ascii grid file, which can for instance be obtained by interpolation of the contours of the orienteering map (Arnet, 2001). To find the fastest route from control A to control B, *LeastCostPath* overlays the relevant part of the orienteering map with an evenly spaced grid of nodes, with each node connected to the 16 closest neighboring nodes, as shown in Figure 2-3. The weight of a connection between two nodes is calculated as the time cost according to the speed model. In this manner, finding the fastest route from control A to control B is reduced to the task of locating the two nodes that are closest to A and B, and applying Dijkstra's algorithm.

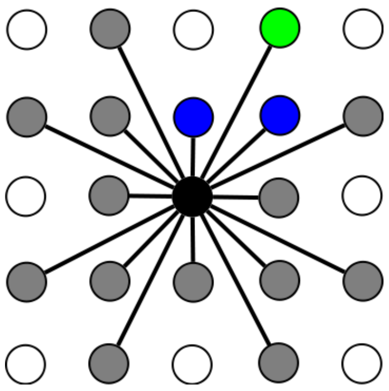


Figure 2-3: Grid neighborhood. The neighborhood of nodes employed in searching for the least cost path in **(Arnet, 2009)**. Consider the task of calculating the distance from the center node to the green node in the upper right. If the nodes were only eight-connected, the shortest path (in terms of distance) to the green node would have to go via one of the blue nodes, leading to a worst-case distance error of 8.0% (5.4% on average). The inclusion of the eight nodes one "knight distance" from the center (named after the moves allowed by a knight in Chess) reduces the worst case distance error to 2.7%, and the average error to 1.3%. See **(Arnet, 2009)** for a deduction of these errors.

If one only desired to find the fastest route from A to B, the algorithm could be stopped once B had been reached. But an orienteer is typically interested in seeing not only the optimal route, but also a selection of other nearly optimal routes in order to compare them. Therefore, the algorithm proceeds until the cost of travelling from A to *any* node in the grid has been computed. The result is a two-dimensional array of costs corresponding directly to the grid of nodes. This array is called the *time model,* and each entry in the array specifies the cost of travelling from A to the corresponding grid point P, assuming the fastest route is taken from A to P. A similar model, called the *inverse time model*, is computed as the cost of travelling from any grid point P to the end node B. Finally, the *route model* is computed as the sum of the *time model* and the *inverse time model*. For each node, it specifies the cost of getting from A to B via that node, assuming that the fastest route is taken from A to P, and from P to B. Figure 2-4 shows an example of a route choice leg, as well as a visualization of the time model, the inverse time model and the *route model*. Instructions on how to interpret the visualizations are given in the figure caption.

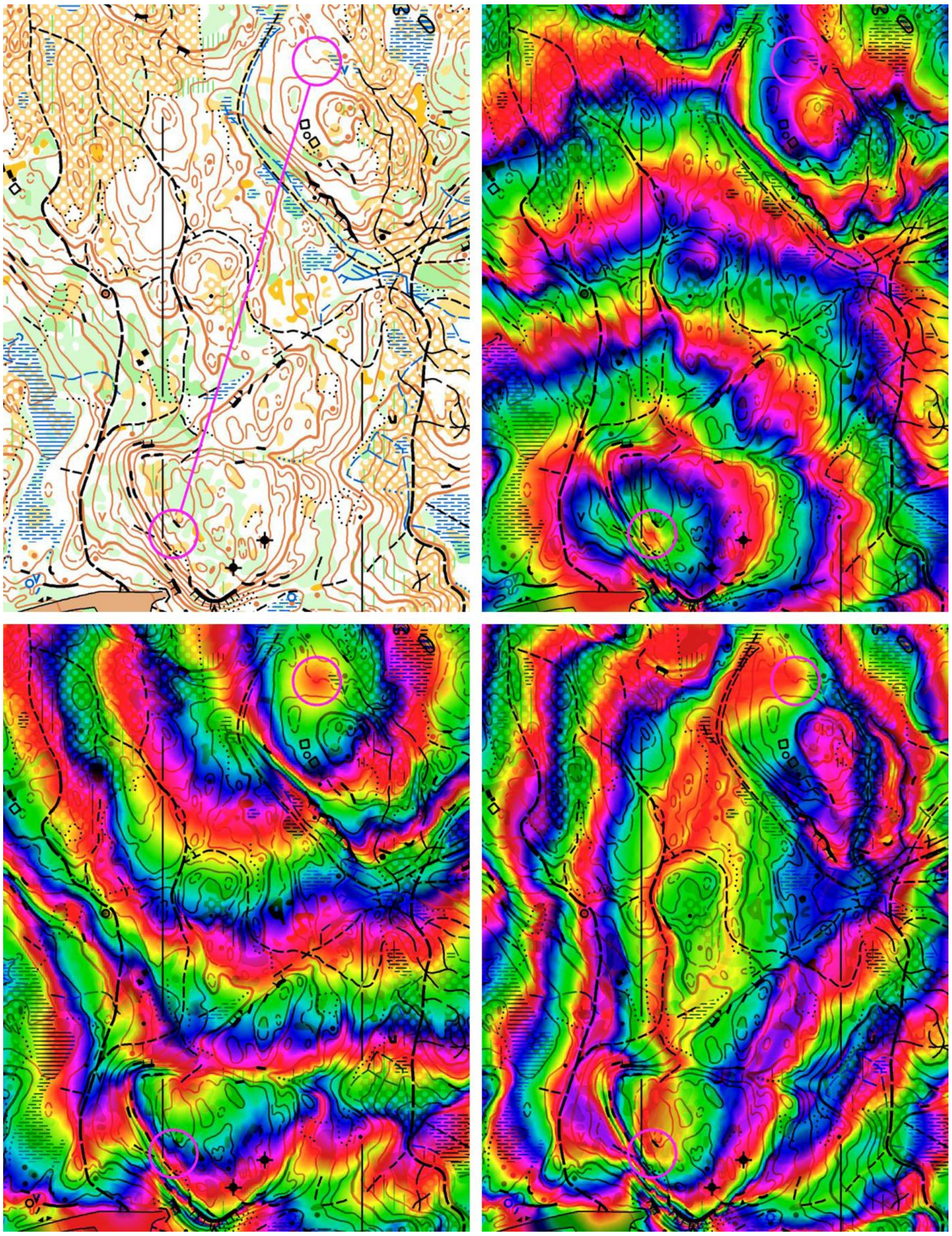


Figure 2-4: Time and Route models. These visualizations were made with the LeastCostPath application, and then superimposed on the orienteering map in OCAD. The top left image shows a route choice leg with the start control in the south and the end control in the north. In the following, the start and end controls will be called A and B respectively. The top right image shows the time model for the leg. For each point P on the map, it specifies the time required to get from A to P. Thus points contained in the same "ring" of constant color around A are equally far from A in terms of the time required to get there. One transition from red to red corresponds to the time required to run 256 meters on a flat road. The lower left image shows the inverse time model. For each point P on the map, it specifies the time required to get from P to B. Thus the value at B in the time model equals the value at A in the inverse time model. The lower right image shows the route model, which is

the sum of the time model and the inverse time model. For each point P on the map, it specifies the time required to get from A to B via P, assuming that the fastest route is taken from A to P, and from P to B. Possible routes are indicated by bands of constant color, where on both sides the color "increases" (red→yellow→green→turquoise→blue→purple→red→...). Such bands of color can be interpreted as valleys in a cost-surface. The higher you climb on the cost-surface, the more time you will have lost, so it is preferable to stay in the lowest valley possible. The red color band that stretches from A to B in the image is this lowest valley, and thus the optimal route according to the applied speed model lies along its bottom. Note that if a runner should stray from the optimal route choice and end up in a color band where on one side the color increases, while on the other side it decreases, he is on a slope of the cost surface and should make his way down to the nearest valley. If he ends up in a color band where on both sides the color decreases, he is on a ridge of the cost surface, and he can choose to proceed down to the valley on either side.

3 Method and Implementation

This chapter describes the proposed method for modeling the running speed in orienteering based on terrain features, as well as the implementation of the method into a working system.

Felix Arnet, who has done extensive work on arithmetic route choice analysis (Arnet, 2009), kindly provided the source code for his application called *LeastCostPath* (see Section 2.4.2) for the purposes of this thesis. Several classes and functions from the included APIs were, with some modifications, used for processing map data, DEMs and GPS tracks into a usable form. In addition the LeastCostPath application itself was modified to use a trained ANN (committee) as a speed model in place of the standard speed model, which was treated in Section 2.4.1

3.1 Proposed Model and Method

On a high level, the proposed method for creating a speed model is as follows:

1. Compile a training set (or multiple sets) of training data from GPS tracks, orienteering map and elevation model. Two consecutive track points are needed to make a training example. The coordinates of each track point are used to look up and extract the relevant features from the orienteering map and elevation model. Each resulting training example contains the present map features, the running slope (i.e. slope in running direction), the terrain slope (the magnitude of the altitude gradient), and finally the running speed.

2. Use the training set to train an ANN using Backpropagation. A separate validation set can be used, or a validation set can be created by randomly extracting a fraction of the training set. Optionally, a test set can also be used to test the ANN after the training has been completed.

There are some important assumptions inherent in this procedure. First of all, running speed in a particular point and direction is assumed to be a function purely of the terrain features that can be read off the map at that point, i.e. the map features, and of the running slope and terrain slope (which can essentially be read off the map's contours). This was also assumed by (Arnet, 2009), as described in Section 2.4.1. There are certainly other factors that can have an impact on the running speed, such as fatigue and not least orienteering mistakes and hesitation, but they are not given much attention in this thesis. To a certain extent, excluding them seems justifiable. This thesis makes use of the observed speeds of the best orienteers in the World Orienteering Championships, as well as those of a test person running in the same terrain, as the basis for learning speed models. It seems reasonable to assume that the best orienteers in the world are able to balance their speed and energy consumption through an entire race, as well as avoid any serious mistakes. As for the test person, the data collection was performed in a controlled situation, where the assigned course was designed to be fairly simple to navigate through, thus minimizing the chance of hesitation and mistakes, and the running sessions were short enough that fatigue should not be an issue. If mistakes had been more common in the data, steps could have been taken to filter the relevant points out of the GPS tracks. Sagberg (2012) discusses some possible approaches for doing this. However, this was not considered necessary for this thesis, due to the reasons just discussed.

One could argue the fact that the risk of mistakes is higher in for instance dense forest than in regular forest, and that this risk can be seen as inherent in the speed predictions made by a model that was trained on data containing mistakes. In this regard, one could say that the speed models produced in this thesis do take the risk of mistakes into account. This seems however to be somewhat oversimplifying, as the risk of mistakes likely depends not only on the terrain features at the current location, but also on the interplay of various terrain features in an area. For instance, an area with lots of small variations in altitude, which on the map would appear as a very detailed contour image, is more technically challenging to navigate through than an area of slowly varying altitudes. This cannot be modeled by looking only at the running slope and terrain slope at the runner's current location.

3.2 System Overview

An ANN-based system for the training of speed models was implemented in the C# programming language, relying on the AForge.NET API (Kirillov, 2012) for the neural network logic. The implemented system has two main functions from the user's point of view:

1. Creation of a training set from one or more GPX files (GPS tracks), an orienteering map and a DEM.

2. Training an ANN, or a committee of ANNs, on a training set.

Figure 3-1 shows a schematic overview of the implemented system, while Figure 3-2 shows a screenshot of the application.

In addition to the implementation of the ANN learning system, additions and modifications were implemented in Felix Arnet's *LeastCostPath* application (described in Section 2.4.2), in order to make it usable with the ANN Speed Predictor. The following subsections detail how the implemented ANN learning system works, as well as how a trained ANN (or a committee of ANNs) can be used as the speed model in the calculation of routes in *LeastCostPath*.

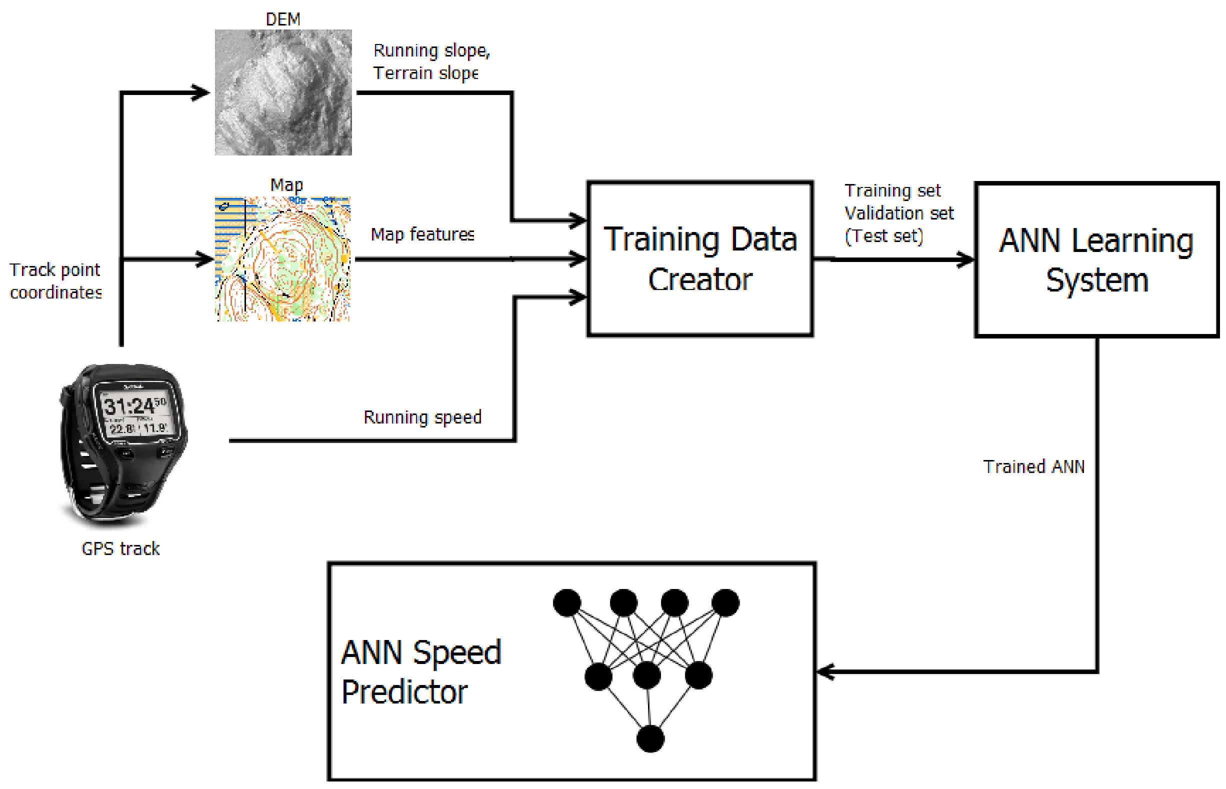
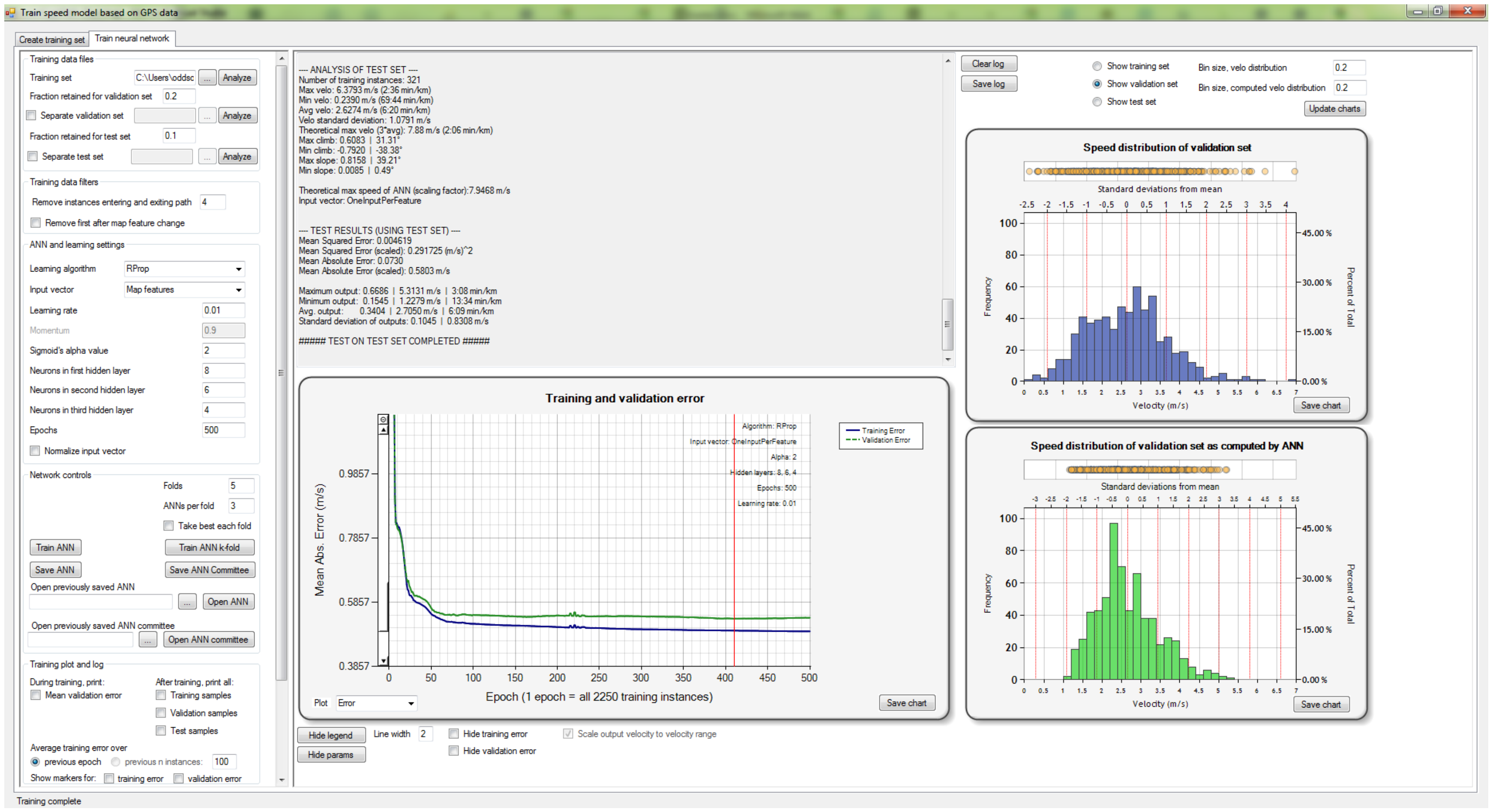


Figure 3-1: ANN Learning System overview. Track points from the GPS data are used to look up the relevant map features, running slope and terrain slope in the orienteering map and DEM. This information, combined with the running speeds deduced from every two consecutive track points, makes up the set of training examples that map terrain features to running speed. The training set is used to train an ANN, which is then stored in binary form and can later be loaded into the ANN Speed Predictor module.



Train speed model based on GPS data

Create training set | Train neural network

Training data files
- Training set: C:\Users\oddsc
- Fraction retained for validation set: 0.2
- Separate validation set
- Fraction retained for test set: 0.1
- Separate test set

Training data filters
- Remove instances entering and exiting path: 4
- Remove first after map feature change

ANN and learning settings
- Learning algorithm: RProp
- Input vector: Map features
- Learning rate: 0.01
- Momentum: 0.9
- Sigmoid's alpha value: 2
- Neurons in first hidden layer: 8
- Neurons in second hidden layer: 6
- Neurons in third hidden layer: 4
- Epochs: 500
- Normalize input vector

Network controls
- Folds: 5
- ANNs per fold: 3
- Take best each fold

Training plot and log
- During training, print: Mean validation error
- After training, print all: Training samples, Validation samples, Test samples
- Average training error over: [✓ previous epoch] previous n instances: 100
- Show markers for: training error, validation error

Bin size, velo distribution: 0.2
Bin size, computed velo distribution: 0.2
[✓ Show validation set]

Line width: 2
[✓ Scale output velocity to velocity range]

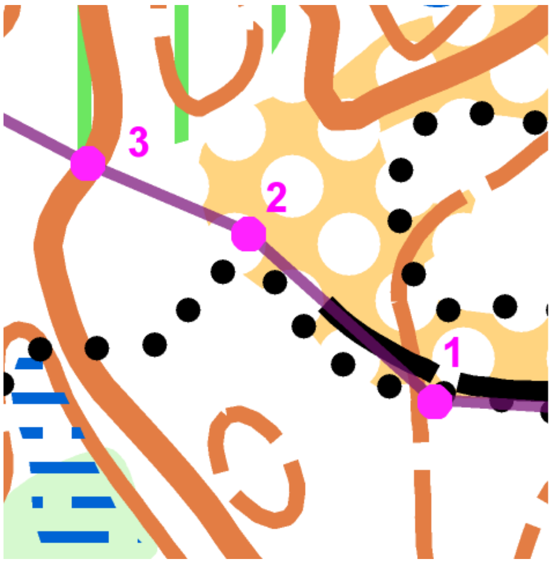
Training complete

```
---- ANALYSIS OF TEST SET ----
Number of training instances: 321
Max velo: 6.3793 m/s (2:36 min/km)
Min velo: 0.2390 m/s (69:44 min/km)
Avg velo: 2.6274 m/s (6:20 min/km)
Velo standard deviation: 1.0791 m/s
Theoretical max velo (3*avg): 7.88 m/s (2:06 min/km)
Max climb: 0.6083 | 31.31°
Min climb: -0.7920 | -38.38°
Max slope: 0.8158 | 39.21°
Min slope: 0.0085 | 0.49°

Theoretical max speed of ANN (scaling factor):7.9468 m/s
Input vector: OneInputPerFeature


---- TEST RESULTS (USING TEST SET) ----
Mean Squared Error: 0.004619
Mean Squared Error (scaled): 0.291725 (m/s)^2
Mean Absolute Error: 0.0730
Mean Absolute Error (scaled): 0.5803 m/s

Maximum output: 0.6686 | 5.3131 m/s | 3:08 min/km
Minimum output:  0.1545 | 1.2279 m/s | 13:34 min/km
Avg. output:      0.3404 | 2.7050 m/s | 6:09 min/km
Standard deviation of outputs: 0.1045 | 0.8308 m/s

##### TEST ON TEST SET COMPLETED #####
```

Figure 3-2: Screenshot of the implemented ANN learning system

3.3 Compiling the Training Data

The basis for the training data are GPS tracks of orienteers, combined with a digital orienteering map and digital elevation model (DEM). However, these raw data must be preprocessed and compiled into an appropriate configuration to be used as training data for the ANN. A program was written to perform the compilation of training data from the raw data sources. The program can load a number of GPS tracks, as well as the orienteering map and DEM (all in specific formats) and compile a training file consisting of all training examples derived from all the GPS tracks. The training file is exported as XML. Figure 3-3 illustrates three consecutive track points from a GPS track properly aligned with the orienteering map, and the training examples that were derived from them. The following subsection explain how the program compiles the training data.



```
   <TrainingInstance>
     <RunningSlope>-0.052389065083746841</Climb>
     <TerrainSlope>0.17700773797015945</Slope>
     <MapFeatures>
1      <Forest>RoughOpenScatteredTrees</Forest>
       <Water>None</Water>
       <Undergrowth>None</Undergrowth>
       <Paths>LessDistinctSmallPath</Paths>
     </ClassifiedMapFeatures>
     <Speed>3.7296976471226095</Velocity>
   </TrainingInstance>

   <TrainingInstance>
     <RunningSlope>-0.045660990328262985</Climb>
     <TerrainSlope>0.12260972458306067</Slope>
     <MapFeatures>
       <Forest>RoughOpenScatteredTrees</Forest>
2      <Water>None</Water>
       <Undergrowth>None</Undergrowth>
       <Paths>None</Paths>
     </ClassifiedMapFeatures>
     <Speed>4.6799636573660859</Velocity>
   </TrainingInstance>

   <TrainingInstance>
     <RunningSlope>-0.0557827353888339</Climb>
     <TerrainSlope>0.069197477542813066</Slope>
     <MapFeatures>
       <Forest>EasyRunning</Forest>
3      <Water>None</Water>
       <Undergrowth>SlowRunning</Undergrowth>
       <Paths>None</Paths>
     </ClassifiedMapFeatures>
     <Speed>3.129335107558544</Velocity>
   </TrainingInstance>
```

Figure 3-3: Deriving training examples from GPS track points, map and DEM. The left part of the figure shows three consecutive GPS track points superimposed on the orienteering map. (The DEM is not shown.) The right part of the figure shows the corresponding entries in the training set file. Climb and slope are computed from the DEM, while the map features are determined by looking at the appropriate coordinates of the orienteering map. Note that *Speed, Running Slope* and *Terrain slope* are based on two consecutive track points. For instance the speed for training example 2 is the average speed between track points 1 and 2. Similarly the running slope of training example 2 is the altitude difference between track points 1 and 2 divided by the distance between those points. Finally the terrain slope of training example 2 is the average terrain slope of track points 1 and 2. The map features are discrete values that cannot be averaged, so the map features of training example 2 are simply the map features present at track point 2.

3.3.1 GPS Tracks

A GPS track in the form of a GPX file provides a set of consecutive track points, recorded with a portable GPS logger, typically worn on the athlete's back or wrist. For each track point, the log file provides a timestamp and the coordinates in latitude and longitude. From this data, the average speed between every two consecutive track points is computed. If the time interval between consecutive track points is sufficiently short, it is reasonable to assume that the terrain, and thus the running speed, does not change considerably in that short time. Thus, the calculated speed between consecutive track points *a* and *b* is taken as the speed in *b*. Finally the latitude and longitude coordinates of the GPS track points are transformed into the UTM coordinate system used for geo-referencing the map. For details on this, see (Hager, et al., 1989).

3.3.2 Orienteering Map - Extracting and Representing Map Features

To determine the terrain features at each track point in the GPS track, it is necessary to look at the appropriate coordinates of the digital orienteering map. The map is geo-referenced, but because it is a vector drawing, it cannot easily be "queried" for the map symbols present at a certain point. (This would involve complicated inside/outside tests on polygons and area borders constructed from Bézier curves.) Instead, the map is converted into a set of geo-referenced raster images, with different shades of gray encoding the presence or absence of various map symbols. (The raster images are exported from OCAD (OCAD, 2012) using a custom symbol set.) Figure 3-4 illustrates this encoding. The orienteering map and the exported raster images are geo-referenced in the UTM coordinate system. The UTM transforms of the coordinates of a GPS track point are used as indices to look up the correct pixel in each raster image. A method for handling this by nearest neighbor interpolation was provided in the class libraries included with the LeastCostPath application. The pixel values retrieved from the set of raster images are real number between 0 and 1, and these number are then interpreted by the Training Data Creator program (see Figure 3-1), which assigns the correct map symbols to the training example according to the encoding illustrated in Figure 3-4.

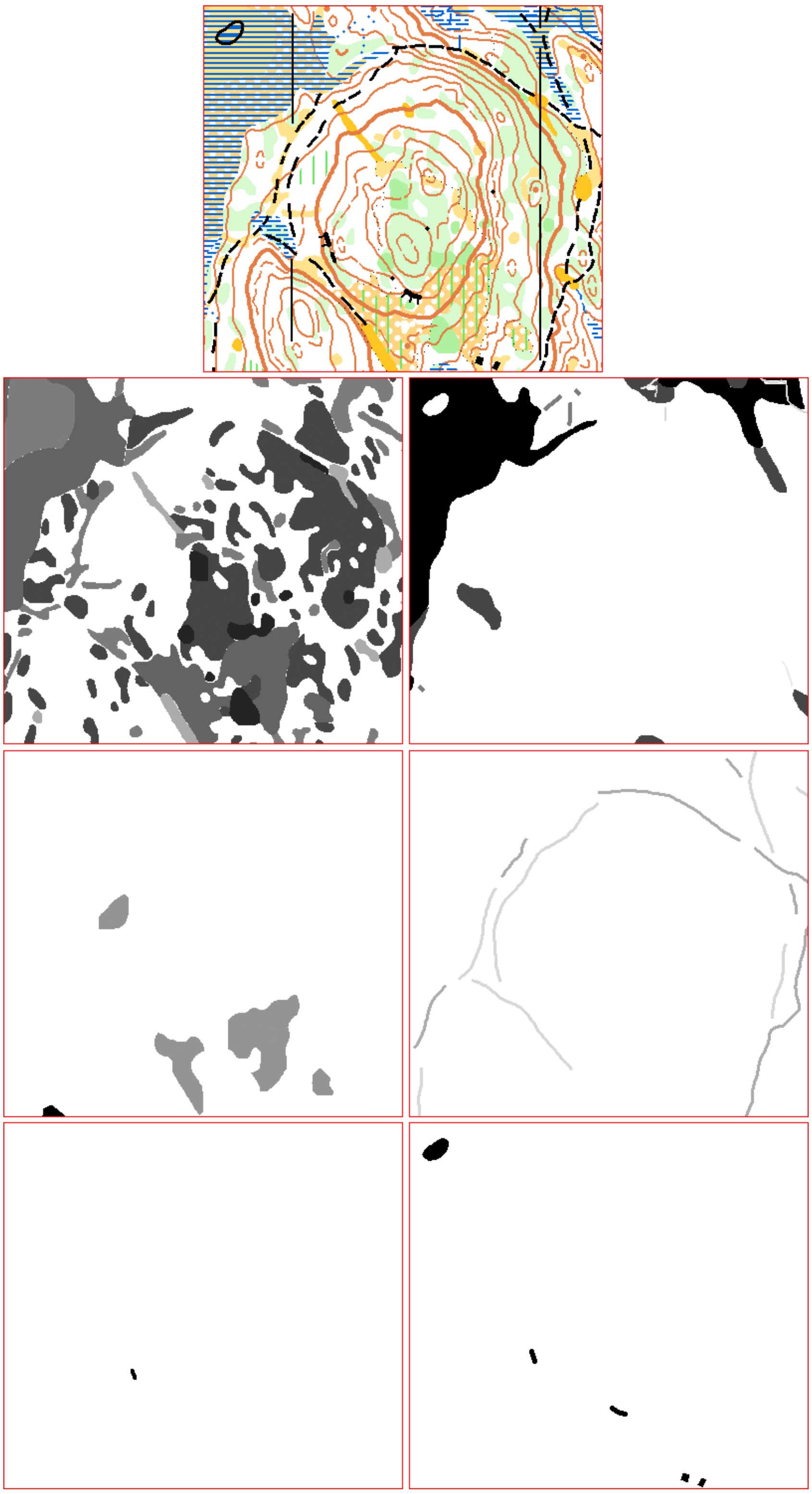


Figure 3-4: Encoding of terrain features. The top image shows the actual orienteering map. The remaining six images show the raster encodings of map feature groups (left to right, top to bottom) forest, water, undergrowth, paths, ground and passable. These groups are described below. Different shades of gray encode different map features of the respective groups.

The orienteering map consists of a large number of different map features, describing various features of the terrain. When devising a representation of the training data, the following points are worth having in mind:

1. Not all map features are relevant to running speed.

2. Certain map features cannot co-occur with certain other map features (see (Persson, et al., 2000)).

Based on these considerations, the map features were divided into six groups, as shown in Table 1**Error! Reference source not found.**. Some of the map features in are not used directly during training. Most importantly, if a training example occurs in an area that is marked as forbidden, that training example should be disregarded during training, since we are not interested in learning the speed in forbidden areas, which should rather be enforced to zero.[1] The following subsections give a brief explanation of the different map feature groups.

Table 1: Groups of map features. Map features written in gray text are implemented as part of the learning system, but they are ignored in the experiments of the next chapter, due to reasons explained below.

Forest	**Water**	**Undergrowth**	**Paths**	**Ground**	**Passable**
Very difficult to run	Marsh	Difficult to run	Road/paved area	Boulder field	Forbidden
Difficult to run	Indistinct marsh	Slow running	Vehicle track	Stony ground	Forbidden, but paths allowed
Slow running	Narrow marsh		Footpath	Broken ground	Passable
Easy running	Pond		Small path	Passable rock face	
Rough open land w/scattered trees	Crossable water-course		Less distinct small path	Earth bank	
Rough open land	Crossable small watercourse		Narrow ride		
Open land w/scattered trees	Minor water channel				
Open land					
Open sandy ground					
Bare rock					

[1] Slight inaccuracies in the GPS data or the border of the forbidden area on the map may lead to training instanc-

Forest

These are area features that mainly describe the density of vegetation. There are three degrees of dense forest (drawn in different shades of green on the orienteering map), and four degrees of more or less open land (drawn with different shades of yellow). In addition there is *open sandy ground* (yellow with a black dot pattern), *bare rock* (gray), and finally regular forest (white on the map), which is neither dense nor open. This is denoted as *easy running*, and is implied when no other forest feature is present. *Open sandy ground* and *bare rock* are ignored in this thesis, as they are virtually non-existent in the test terrain.

Water

This category comprises everything wet, essentially marshes and streams. Only marshes are used for training the speed model. Watercourses and water channels are actually quite frequent, but orienteers typically don't run in them (they just cross them), so the amount of relevant training data for these features is expected to be low. Given their small width, it can be assumed that the time cost of crossing a watercourse/water channel is negligible. Lakes and wide rivers belong to the impassable category, as swimming is not allowed in orienteering. For this thesis, *indistinct marsh* and *narrow marsh* are considered as the same map feature from the ANN system's point of view.

Undergrowth

If an area is covered with dense vegetation that only hinders runnability, but not sight, the undergrowth map features are used. There are two degrees of undergrowth density, indicated by differently spaced green hatch patterns. Undergrowth can be combined with the various degrees of open forest or regular forest (i.e. on top of yellow and white).

Paths

Paths and roads go into this category. For this thesis, six variants of paths/roads are used. On the actual orienteering map there are more road types, but it can be safely assumed that running speed on a 10 m wide road will not be significantly different from that on a 3 m wide road. In addition, *vehicle track* is considered equal to *road* from the ANN system's point of view because they are essentially the same as small roads in the test terrain. (It is possible that it would be preferable to change this in a different terrain.) The paths are represented as line objects in the digital orienteering map. In order to account for small inaccuracies in the GPS data and possibly the positioning of the paths on the map, the paths and roads are given exaggerated width in the raster map that is used to create the training data. The reasoning behind this is simply that an orienteer is far more likely to be running on a path/road than right next to it, which he may sometimes appear to be if one trusts the GPS data completely. This does however introduce problems when the orienteer is merely crossing a path, and when entering and exiting a path, since too many track points are then classified as being on the path. This is alleviated by filtering out the first few track points after entering a path and the last few before exiting.

Ground

These are features that describe the running surface. The most important ground feature which affects running speed is rocky ground. Other features include passable rock faces, earth banks and broken ground. This category of map features presents a few problems. Stony ground is difficult to rasterize into a bounded area, as it is drawn on the map as a group of point objects. The amount of rocks is indicated by the density of these objects. Passable rock faces are also tricky, as they are quite small in area. Because of the slightly inaccurate GPS track, it will be very difficult, if not impossible, to tell whether a runner climbed a small rock face or whether he simply passed close by. For these reasons, and because most of the ground features (except for passable rock faces) are quite rare in the test area, the ground category is ignored in this thesis, although the implemented system supports it.

Passable

Some features are defined as impassable, i.e. the orienteer cannot go through them. This includes buildings, private property, cultivated land, lakes and rivers, dangerous swamps etc. For some of these areas, it is possible that a road or path passes through (a bridge in the case of rivers), and in such cases it is usually allowed to pass through on the road/path. As mentioned in the introduction of this section, this group of map features is not used for training the ANN, but rather for overruling the ANN's speed predictions with a speed of 0.

3.3.3 Digital Elevation Model - Extracting Slope Information

The digital elevation model DEM is a grid that specifies the elevation (altitude) of the terrain at evenly spaced coordinates. It can be obtained by two different means: From laser scans of the terrain or by interpolating between the contours of the digital orienteering map, as described by (Arnet, 2001). Figure 3-5 illustrates the difference between the two approaches. As should evident from the figure, the laser based DEM contains far more fine detail than the contour interpolated DEM. (The implications of this are discussed below.) No matter which alternative is chosen, the resulting DEM is an evenly spaced grid of real values, detailing the altitude (in relation to some base level) of the terrain at each grid point. The DEM is georeferenced in UTM, like the orienteering map, so the correct altitude for a certain GPS track point can be found by bilinear interpolation between the nearest grid points in the DEM. A method for handling this was provided in the class libraries included with the LeastCostPath application.

But indexing to the DEM grid point that corresponds to a GPS track point only yields the altitude at that point. What we need is actually the *running slope* and the *terrain slope*. Running slope, i.e. slope in the running direction, is computed as the altitude difference between two consecutive GPS track points divided by the straight line distance between the two points. This number theoretically ranges from negative infinity to positive infinity, but typically stays approximately in the range of $(-1, 1)$, which corresponds to the angle range of $(-45°, 45°)$. The terrain slope is the magnitude of the altitude gradient, i.e. the slope in the direction of steepest ascent, which is given by

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \textbf{(3.1)}$$

where $f(x,y)$ is the altitude of the terrain at coordinates (x,y). The partial derivatives in x and y direction are also computed by bilinear interpolation between grid points and the point of interest. A method for computing the gradient magnitude in this fashion was provided in the class libraries included with LeastCostPath. The gradient magnitude theoretically ranges from 0 to positive infinity, but typically stays in the range $(0, 1)$, i.e. $(0°, 45°)$. (As an alternative to the terrain slope, we could also have used the *cross slope*, which can be defined as the slope of a vector parallel with the surface at the runner's position and perpendicular to the direction of travel.)

An important consideration is that of the DEM's accuracy compared to the accuracy of the GPS track points. In the experiments conducted for this thesis, a laser-scanned DEM was used. The altitude for a given grid point in the DEM is accurate to a few centimeters. Compare this with the GPS accuracy of a few meters, and a possible pitfall reveals itself. Using a DEM with a grid size of 1m x 1m to compute the terrain slope at the coordinates of a GPS track point may give a misleading result if the track point is a few meters off. Due to the fine detail of the DEM, the terrain slope at the offset point may be quite different from the terrain slope at the runner's actual position. The running slope is also affected, but to a lesser extent. To smooth out this problem, a DEM with a larger grid size of 3m x 3m was used.

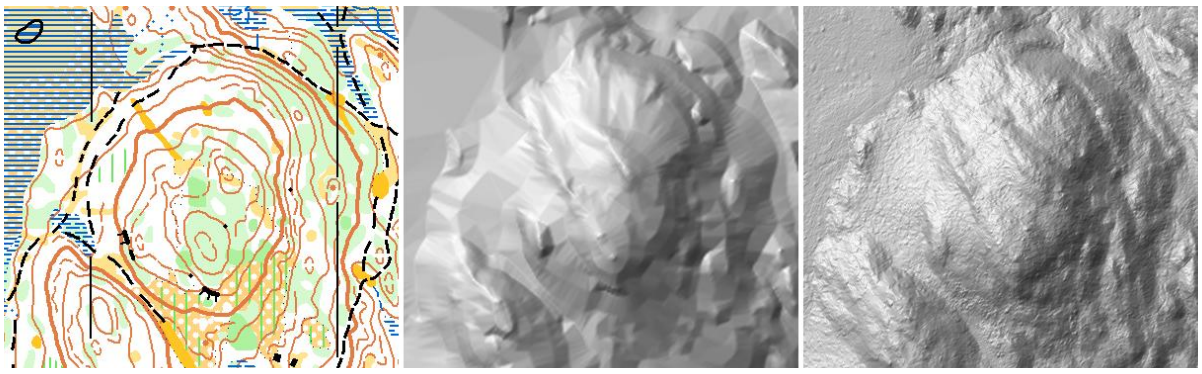


Figure 3-5: Two alternatives for obtaining a DEM. The top image shows the orienteering map. The two other images show hill shaded relief renderings of the two DEM alternatives. The first was obtained by contour interpolation, courtesy of Felix Arnet. The second was obtained from laser scans of the terrain. The renderings were created using OCAD (OCAD, 2012).

3.4 Constructing the Input Vector

A multi-layer ANN accepts as input a vector of real values and outputs a vector of real values. For the speed model, there is just one output, the running speed. The input vector however, must fully describe the terrain features that lead to the output running speed. Intuitively, we could use one network input for each of the map feature groups discussed in Section 3.3.2 (except for the Passable group and the Ground group, as explained previously), and assign a real value, or a range of real values, to encode each map feature of the group. Still, a simpler solution is to create one input for each possible map feature, and simply set the value to 0 or 1 depending on whether the map feature is present or not. Both approaches were implemented. An illustration of a network using the grouped map features approach can be seen in Figure 3-6.

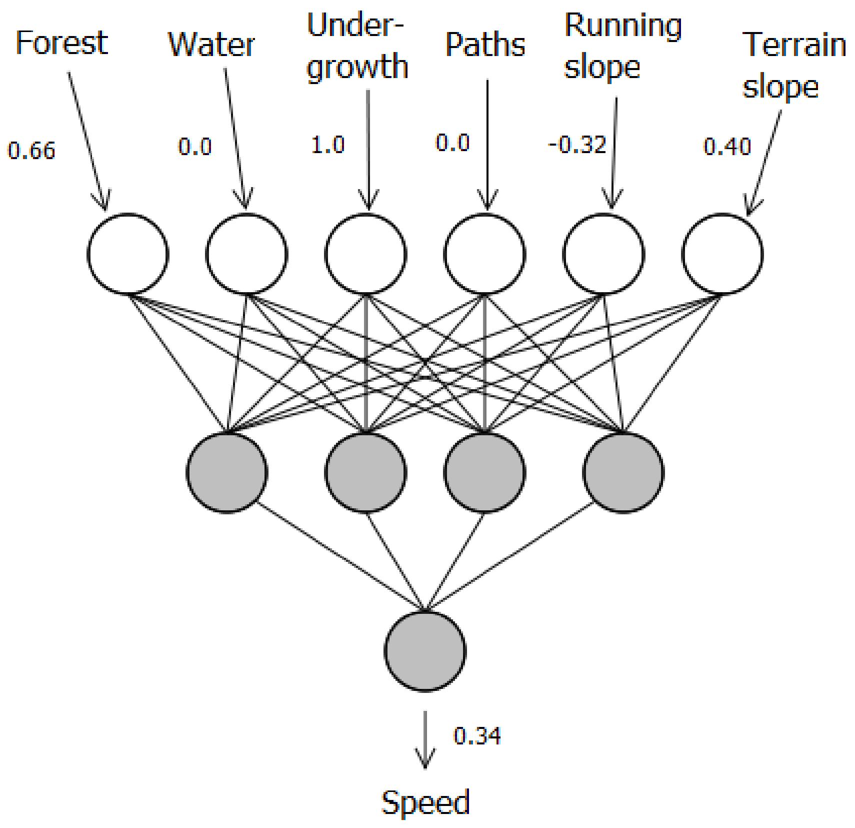


Figure 3-6: Example ANN speed model. This ANN uses the grouped map features input vector and one hidden layer with four neurons. Example values for the different inputs as well as the output are shown. Note that the output speed is relative to a theoretical maximum speed. It is scaled into the real speed range by multiplication with the theoretical maximum speed. (See Section 3.5.)

As discussed in Section 2.3.1, normalization of the input vector is an important issue to consider. However, because all the inputs of the input vector implemented in this thesis are usually in the $(0,1)$ range (true for both input vector types just discussed), this was not deemed necessary. The map feature inputs are always in the $(0,1)$ range. The terrain slope can theoretically reach positive infinity, but is usually also in the $(0,1)$ range. The running slope, however, is usually in the $(-1,1)$ range. This was not considered so large a deviation from the other inputs that normalization was considered necessary.

3.5 Neuron Activation Function

The Sigmoid function was used as the activation function in all hidden layers and the output layer. As discussed in Section 2.3.1, the Sigmoid function has the output range of $(0,1)$. Ideally, a linear activation function should have been used in the output layer in order for the ANN to be able to output the actual running speeds. Unfortunately the implementation of the Backpropagation algorithm in the AForge.NET framework assumes that all the layers of the ANN have the same activation function. Therefore a scaling approach was taken. A theoretical maximum speed is calculated as three times the average speed of all the training examples in the training set with which the ANN is trained. This number appears always to be slightly higher than the maximum observed speed. Before training, the target speeds of the training set are divided by the theoretical maximum speed to scale them into the $(0,1)$ range, and in subsequent use of the trained ANN, the outputs are multiplied by the same factor. Because the lowest observed running speed always is above zero, and the highest speed always is below the theoretical maximum[2], the ANN should never have to output values that are very close to 0 or 1, which would require the weighted sum of inputs to approach negative or positive infinity, respectively. This is however a possible weakness of the implemented system, and it is possible that using a linear activation function in the output layer would lead to better accuracy.

3.6 Tunable Learning Settings

The implemented system provides a wide selection of learning parameters. Backpropagation (Backpropagation and Resilient Backpropagation can both be used as the learning algorithm. The Backpropagation algorithm available in the AForge.NET framework is the online version, in which the weights are updated after each presentation of a training example. The Resilient Backpropagation algorithm, however, runs in batch mode, i.e. it updates the weights at the end of each epoch. Other settings include learning rate, momentum (does not apply to Resilient Backpropagation), the alpha value of the Sigmoid function (see Eq. (2.1)), number of hidden layers and neurons, and the number of epochs to run. In addition, k-fold cross-validation, as discussed in Section 2.3.2, was implemented.

3.7 Testing a Trained ANN

Once an ANN has been trained, it can be tested using a test set. The test set consists of training examples from the same distribution as the training and validation sets. Like the validation set, it can be acquired by extracting at random a small fraction of the training set, or a separate set can be used (i.e. from a separate file).

In addition, the ANN can be tested with a set of *key feature tests*. A key feature test is simply a test case consisting only of input to the network. Thus, a key feature test set can be put together to reveal specific properties of the speed model, such as how the speed varies with the running slope in a specific type of forest. The results will have to be evaluated subjectively.

3.8 Using a Trained ANN for Route Choice Analysis

Felix Arnet kindly provided the source code for his application LeastCostPath, described in Section 2.4.2, for the purposes of this thesis. A number of additions and modifications were made to that application in order to replace the standard speed model (see Section 2.4.1) with an ANN-based model. The trained ANN, or ANN committee if k-fold cross-validation is used, is stored as a binary file, which can be loaded by a separate software module called the ANN Speed Predictor. This module is compiled as a DLL file, which LeastCostPath can load and use as its speed model. The interplay between LeastCostPath and the ANN Speed Predictor is illustrated in Figure 3-7.

[2] The opposite has not been observed during this thesis.

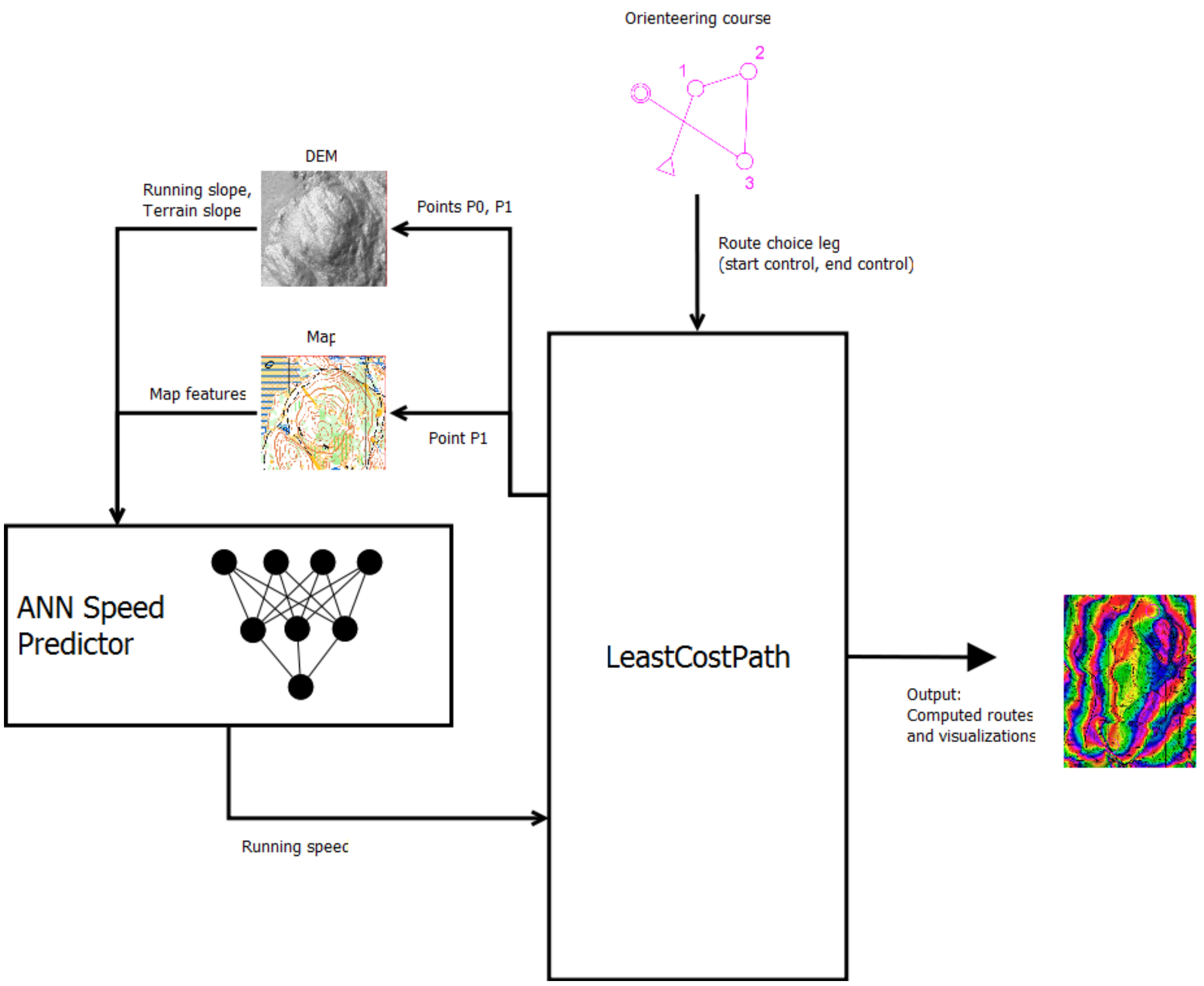


Figure 3-7: Using an ANN speed model for route choice analysis. LeastCostPath runs Dijkstra's algorithm to find the optimal route (and other sub-optimal routes) from one control to the next. The ANN Speed Predictor is used for computing the running speeds that decide the cost of the edges between nodes in the search grid. (P0 and P1 are the two nodes between which LeastCostPath currently needs to know the running speed.)

One important feature which was added to LeastCostPath for this thesis is the calculation of the time cost for a specific route (a GPS track), according to the speed model. This could be done simply by retrieving the terrain features for each track point, computing the cost of each segment between track points and summing up. However, here we face the same problem as when extracting terrain features for the training data: The slight inaccuracy of the GPS track. If the orienteer runs on a path, the GPS track may be slightly off the path. If he passes close by a small area of dense vegetation, the GPS track may go directly through the dense vegetation. In training data creation, the path problem was solved by exaggerating the path widths. This would work here as well, but we can do better.

The implemented solution uses the GPS track to create a "corridor" on the map. (The width of the corridor can be set in the user interface.) Every part of the map except for the corridor is made impassable. Then LeastCostPath is run to find the optimal path through this corridor. This not only forces the a slightly offset GPS track onto a nearby parallel path, but also essentially adjusts the route to avoid any small obstructions the GPS track runs through, but that the orienteer really avoided.

4 Experiments and Results

This chapter describes the experiments that were conducted in order to verify and validate the implemented system and method. This separates naturally into two separate experimental phases. The first phase is verification of the implementation, i.e. verifying that the system implements the suggested method correctly. The second phase is validation of the method itself, in which the implemented system is applied to evaluate the method's usefulness for predicting running speed.

Section 4.1 describes the data, i.e. GPS tracks, map and DEM, that was selected for conducting the experiments. Section 4.2 deals with verification of the implementation. In Section 4.3 the various learning parameters and their impact on convergence and prediction accuracy of the trained ANN are discussed. Finally, Section 4.4 presents the experiments and the results that were conducted. A short discussion of the results is given for each experiment

4.1 Selection of Data for Experiments

When selecting a terrain for testing the method, some criteria had to be considered. First of all, the map had to be of high quality. Since the map would essentially be queried for the map features present at locations where runners had passed, the map must be as precise as possible in order to minimize errors. Second, GPS tracks of elite orienteers running in the terrain should be available. Third, the quality of the GPS tracks had to be considered, i.e. accuracy and sampling rate. Finally, the terrain should not be too technically demanding orienteering-wise, as this tends to increase the amount of mistakes made by the athletes. In such terrains, the second criterion of route choices discussed in Section 2.1 comes into play; the orienteer will not only have to consider which route is the fastest, but also whether that route poses a too high risk of making mistakes. For instance, in the 2011 World Championships in France, the complexity and detail of the maps proved too much of a challenge for many an orienteer. Figure 4-1 shows a tracking screenshot of an orienteer making a severe mistake. For the purposes of this thesis, we are interested in the athlete's running speed when the orienteering work is properly executed, i.e. the technical challenges are not a major hindrance to the speed. In this regard, mistakes introduce noise in the training data.

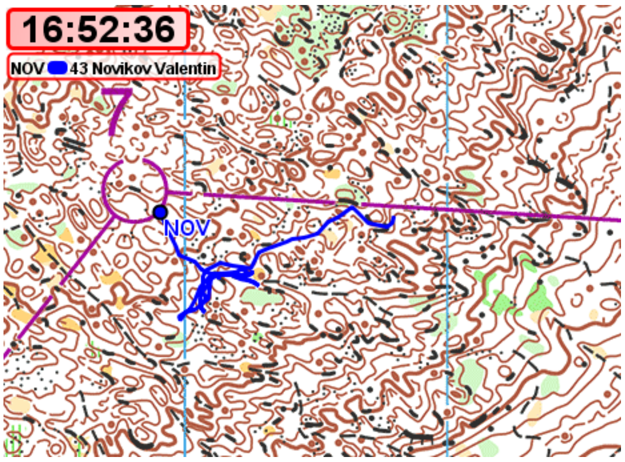


Figure 4-1: Orienteering mistake. Valentin Novikov made a severe mistake on the way to the 7th control of the WOC2011 Middle Distance Final. Source: **(Varis, 2012)**

Based on these criteria, one particular orienteering event stood out as an obvious choice: The 2010 World Orienteering Championships in Trondheim (WOC2010). The maps are of excellent quality, the level of technical difficulty fairly moderate, and the GPS tracks are freely available online. The accuracy of the GPS tracks is generally good (based on visual inspection of tracks superimposed on the map), but the sampling rate of one track point per seven seconds is not ideal, and as the experiments in the next chapter reveal, this was a problem, mostly because it meant that the number of training examples for each runner was quite limited. Unfortunately this sampling rate is typical for the tracking equipment used during such orienteering events, so nothing significantly better was available in this regard. However, fairly inexpensive GPS watches can be used to record tracks with a sampling rate of one track point per second. Such a watch was used by a test subject as part of this thesis to gather a few GPS tracks of higher quality.

The OCAD file (digital orienteering map in vector form) from the WOC2010 finals was acquired from the map owners.

There were essentially two options for acquiring a DEM: Interpolate between the contours of the orienteering map or use a model obtained by laser scans. Felix Arnet kindly assisted in the conversion of contours to DEM, using the method described in (Arnet, 2001). However, due to some anomalies in the resulting DEM, a laser-based DEM was used for the final experiments. The laser data was acquired from Trondheim municipality and converted to the required format using ArcGIS (ESRI, 2012), which is a sophisticated suite of GIS software with a vast number of usage areas, handling of laser data being one of them. The raw laser data is actually a point cloud, and points can be classified as ground, building, water, various classes of vegetation etc. The conversion process consisted of filtering out all points but the ground points, and then use the ground points to create an elevation grid. A grid size of 3m x 3m was used for the DEM.

4.2 Verification of implementation using a small training set

In this section the implemented system is tested to make sure that it correctly implements the intended model, as described in Section 3.1

It is to be expected that ANNs produced by the implemented system will not be able to predict running speed perfectly. The training data consists of a great deal of noise, where several examples with almost equal inputs have different target outputs. This is due to external factors not expressed by the input data, such as variations in runnability which is not represented on the map, hesitation and mistakes by the orienteers, and possibly fatigue. However, given a very small training set without conflicting training examples, a well conceived ANN should be able to learn them perfectly.

Thus, to verify that the implemented system was working as intended, a training set consisting of only three training examples was created. The three examples were selected from a larger set of training examples based on a GPS track collected by a test subject. The examples were selected with the purpose of being non-conflicting and of spanning the whole speed range. Thus the first training example had a very low target speed, the second a very high target speed, and the third had a target speed somewhere in between. The input parameters of the three examples were all different.

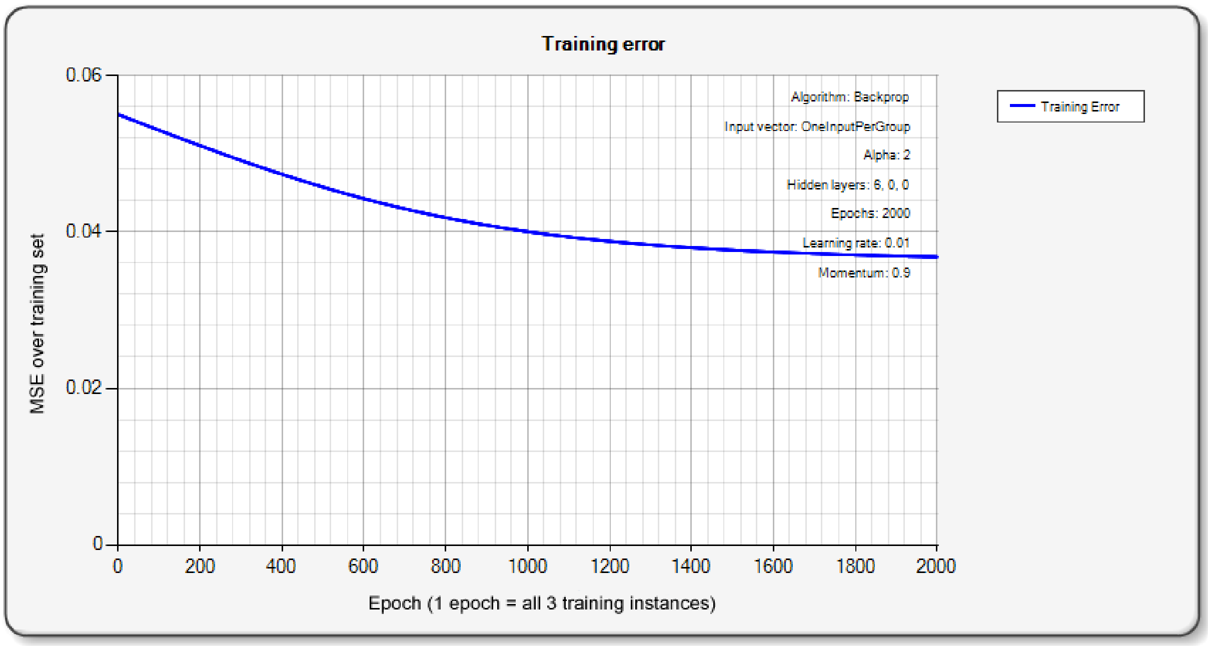


Figure 4-2: Plot of the training error while learning the three test examples, using the unchanged implementation of Backpropagation in the AForge framework. The ANN and learning parameters are given in the top right corner of the chart. "Algorithm" specifies which learning algorithm was used to train the ANN. "Input vector" states which of the two input vector types was used. "Alpha" is the alpha value of the Sigmoid activation function (see Section 2.3.1). "Hidden layers" gives the number of neurons in the first, second and third hidden layer. (A value of 0 indicates that the corresponding layer is not used.) The final three parameters are the number of training epochs, the learning rate and the momentum.

The ANN was constructed with one hidden layer consisting of 6 neurons and trained with a learning rate of 0.01. Momentum was set to 0.9, as suggested by (Hertz, et al., 1991). Other learning parameters can be seen in the top right part of the chart in Figure 4-2. As the figure shows, the training error did decrease throughout training, but very slowly. Running the same training examples through a different Matlab-based ANN application, using similar learning parameters, produced a very different result. The learning process converged to a very low error within a few epochs, which suggested a problem in the implemented system. Inspection of the relevant method in the AForge framework revealed that the Backpropagation algorithm implemented there deviated from the typical formulation. The weight update rule normally applied in Backpropagation with momentum, as stated by (Mitchell, 1997) and others, is

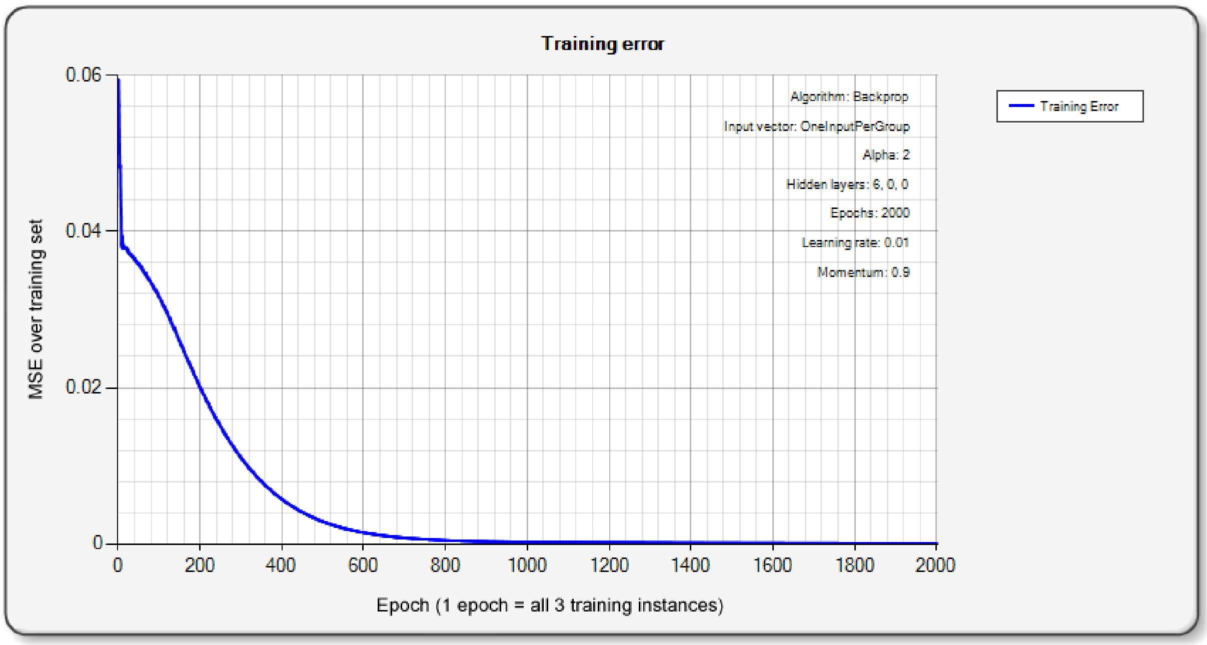
$$\Delta w_{ji}(n) = -\eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1) \qquad \textbf{4.1}$$

Here, $\Delta w_{ji}(n)$ is the weight update of the weight from neuron i to neuron j in learning iteration n, η is the learning rate, δ_j is the error term of neuron j, x_{ji} is the input value to neuron j from neuron i and α is the momentum factor. However, the implementation of the Backpropagation weight update rule in the AForge framework corresponds to the following equation:

$$\Delta w_{ji}(n) = -\eta(1-\alpha)\delta_j x_{ji} + \eta\alpha \Delta w_{ji}(n-1) \qquad \textbf{4.2}$$

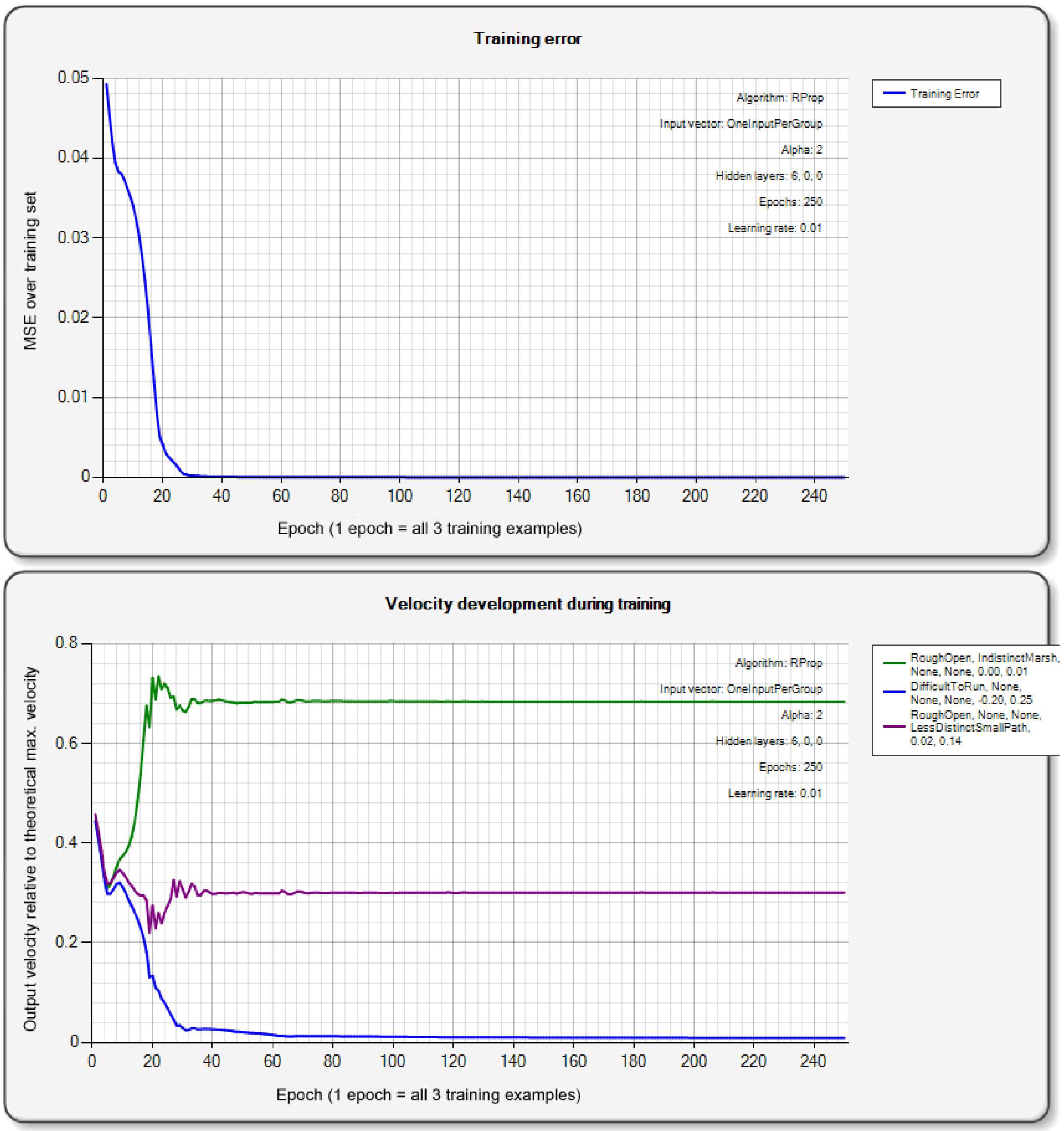
Whereas in Eq. 4.1 the momentum term is simply added to the standard Backpropagation weight update rule, Eq. 4.2 essentially distributes the learning rate between the two terms according to the momentum factor, so that a high momentum factor leads to a very low effective learning rate, and a low learning rate leads to a low effective momentum. For instance, using learning rate 0.01 and momentum 0.9 as above, the effective learning rate becomes 0.001, and the effective momentum becomes 0.009.

Consequently, the AForge implementation was altered to implement Eq. 4.1. Running the same experiment again rendered far better results, as can be seen in Figure 4-3. The learning process converges towards zero error as expected. Using a higher learning rate, such as 0.1, speeds up the process significantly. Additionally, switching to the Resilient Backpropagation algorithm discussed in Section 2.3.3 makes it much faster. This is illustrated in Figure 4-4. In addition to the error plot, this figure also shows the development of the output speeds of the three training examples. Notice how the speeds fluctuate while the error is decreasing, before converging to a steady state.



Algorithm: Backprop
Input vector: OneInputPerGroup
Alpha: 2
Hidden layers: 6, 0, 0
Epochs: 2000
Learning rate: 0.01
Momentum: 0.9

Figure 4-3: Plot of the training error while learning the three test examples, after changing the weight update rule of the Backpropagation algorithm in the AForge framework.



Algorithm: RProp
Input vector: OneInputPerGroup
Alpha: 2
Hidden layers: 6, 0, 0
Epochs: 250
Learning rate: 0.01

Figure 4-4: The top chart shows the training error when using the Resilient Backpropagation algorithm. The bottom chart shows how the output speeds of the three training examples develop during training. The error converges quickly towards zero, while the output speeds converge to constant values.

4.3 *Selection of ANN Settings and Learning Parameters*

In this section, the impact of some of the ANN and learning parameters on the behavior of the learning process is discussed. Throughout the section, examples of error plots and speed plots similar to those of Figure 4-4 are used to illustrate the effect of varying the different parameters. One difference is that the training error and validation error measures plotted in the charts from this point on is the Mean Absolute Error over the training set and validation set, where the absolute error for one training example is simply $|velocity_{output} - velocity_{target}|$. This measure is used because it provides a more intuitive illustration of the actual accuracy of the ANN's speed predictions, even though the squared error is still used to drive the gradient descent search of the Backpropagation algorithm.

Unless stated otherwise, the key feature test set used for speed plots is as follows:

- There are 11 key feature tests where the map features are held fixed (*EasyRunning*, i.e. regular forest), while running slope varies from -0.6 to 0.6. Each key feature test case is listed in Table 2.

- Terrain slope for these tests is the absolute value of running slope. This entails that the runner is going straight up or down the hill along the steepest ascent/descent.

- This test set provides a nice illustration of the ANN's capability to represent non-linear functions, in this case the relationship between running slope and running speed.

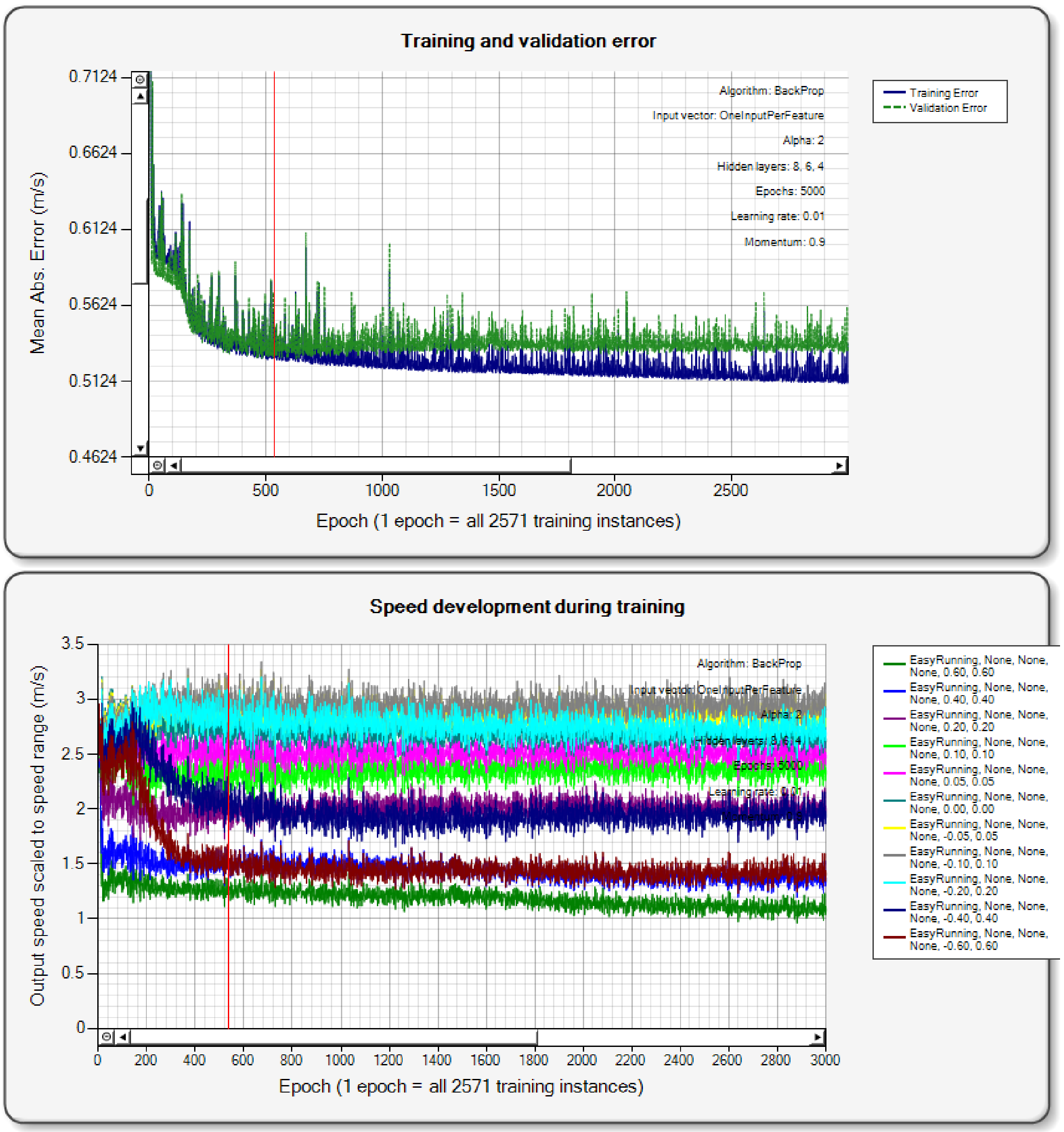
Table 2: Key feature test set designed to reveal the non-linear relationship between running slope and running speed.

Forest	**Water**	**Undergrowth**	**Paths**	**Running slope**	**Terrain slope**
EasyRunning	-	-	-	-0.60	0.60
EasyRunning	-	-	-	-0.40	0.40
EasyRunning	-	-	-	-0.20	0.20
EasyRunning	-	-	-	-0.10	0.10
EasyRunning	-	-	-	-0.05	0.05
EasyRunning	-	-	-	-0.00	0.00
EasyRunning	-	-	-	0.05	0.05
EasyRunning	-	-	-	0.10	0.10
EasyRunning	-	-	-	0.20	0.20
EasyRunning	-	-	-	0.40	0.40
EasyRunning	-	-	-	0.60	0.60

4.3.1 Learning algorithm

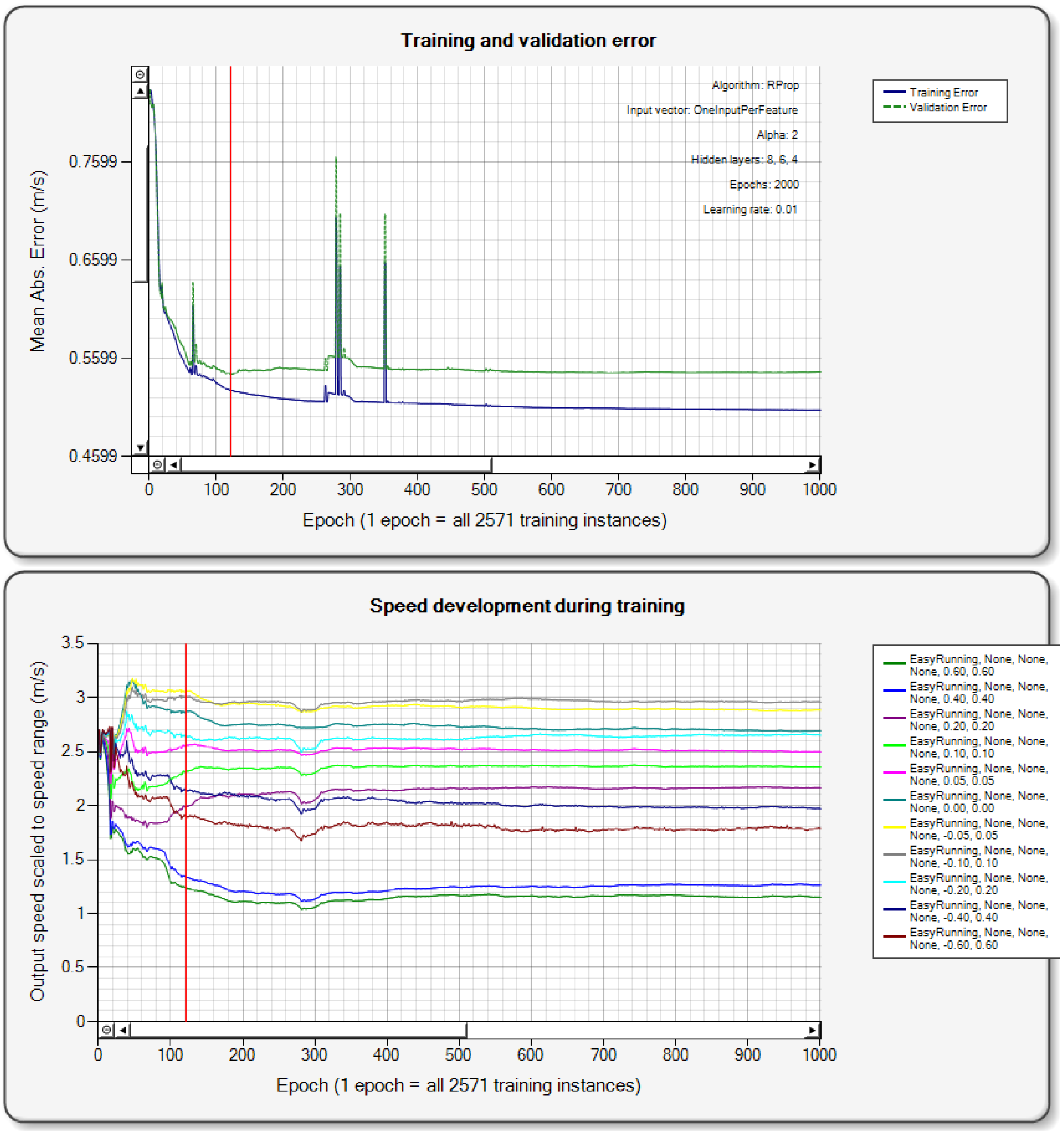
The implemented system provides a selection of two learning algorithms to train the ANN. These are Backpropagation with momentum, hereafter referred to as BackProp, and Resilient Backpropagation, hereafter referred to as RProp. These are both described in Section 2.3.

Figure 4-5 and Figure 4-6 show error curves and speed curves for BackProp and RProp, with all other learning parameters fixed. Even though the scale on the X axis is not the same in all the charts of the two figures, it is easy to spot some interesting differences. First of all, RProp converges to the minimum error over the validation set much faster than BackProp (epoch 83 and 606 respectively). This is to be expected, as speed of convergence is the primary trait of RProp. Secondly, both error and speeds produced by BackProp vary considerably more from one epoch to the next than those produced by RProp. This is likely due to the fact that the size of the weight update in BackProp depends on the magnitude of the error gradient, whereas in RProp the weight update only depends on the sign of the error gradient. Reducing the learning rate for BackProp makes the variations smaller, but the algorithm naturally takes even longer to converge.



Algorithm: BackProp
Input vector: OneInputPerFeature
Alpha: 2
Hidden layers: 8, 6, 4
Epochs: 5000
Learning rate: 0.01
Momentum: 0.9

Figure 4-5: Plots of training/validation error and key feature speeds for BackProp. The vertical red line indicates the epoch where the minimum error over the validation set was observed. The key feature tests are listed in the legend of the lower chart. They are the tests given in Table 2.



Algorithm: RProp
Input vector: OneInputPerFeature
Alpha: 2
Hidden layers: 8, 6, 4
Epochs: 2000
Learning rate: 0.01

Figure 4-6: Plots of training/validation error and key feature speeds for RProp. The vertical red line indicates the epoch where the minimum error over the validation set was observed.

Note that even when the key feature speeds vary considerably from epoch to epoch, they do vary together. Looking at the speed curves over a limited number of consecutive epochs, it is clear that when the slope of one curve is increased (decreased), the slopes of the other curves also increase (decrease). This is true for both BackProp and RProp and is illustrated in Figure 4-7. The practical implication of this is that in the short term (few epochs) the ratios between the key feature speeds remain stable, whereas in the long term they are allowed to evolve into their proper values.

As for accuracy over validation set and test set, the two algorithms are quite similar, given the time to converge. On the basis of what has been discussed here, RProp was used for the final experiments in Section 4.4.

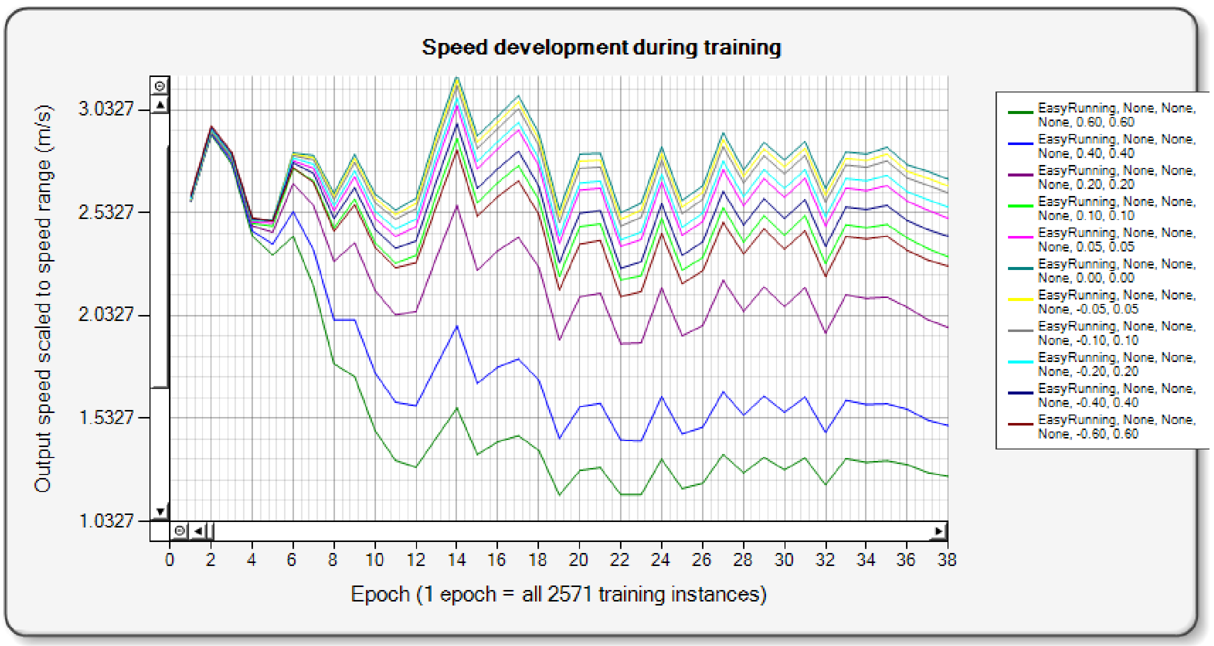


Figure 4-7: Detail of the first epochs of the speed plot in Figure 4-5. The ratios between speeds are stable in the short term, even if the speeds themselves are not.

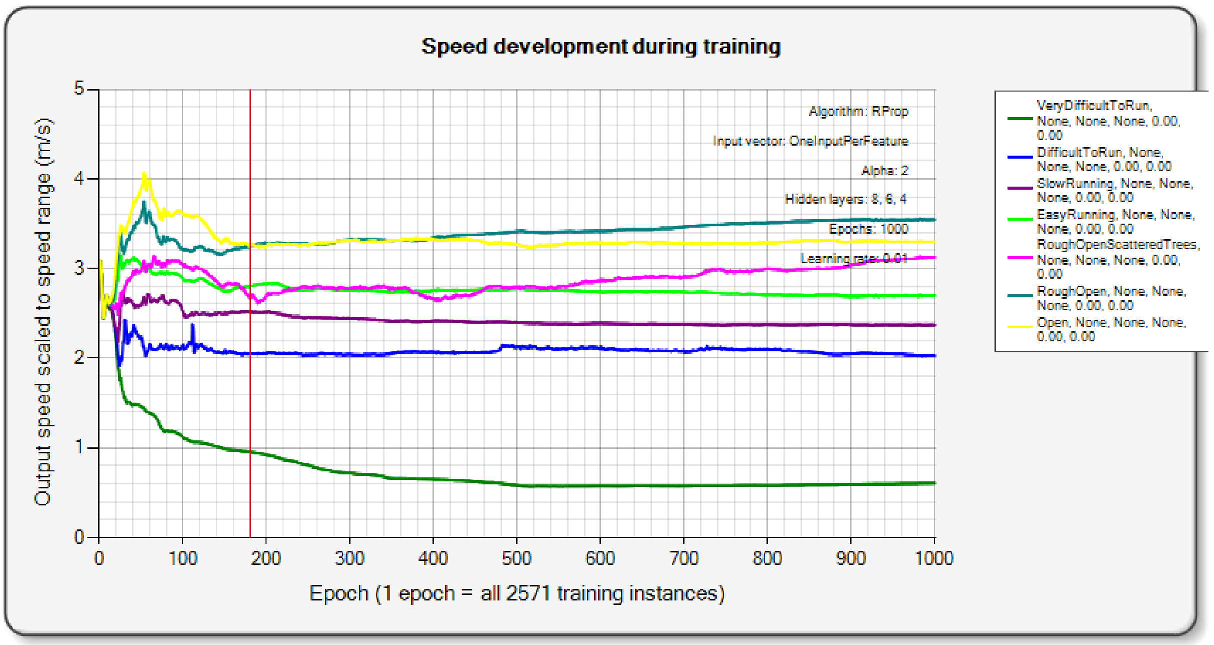
4.3.2 Input Vector

The two implemented input vector approaches differ in the way they represent the map features (see Section 3.4). Figure 4-8 illustrates the practical difference between them. The key feature test set used here consists of seven test cases that only vary in the type of forest. These test cases are listed in Table 3. The first input vector type, which uses one input for each map feature, is the most flexible, as it provides the ANN with the most degrees of freedom to learn the effect of each map feature. The second type, which uses one input for each group of related map features, is somewhat more restrictive, as it encodes implicit knowledge of the correct order of the map features within a group in terms of speed. For instance, the value of the input for the Forest group ranges from 0 for *VeryDifficultToRun* to 1 for *Open*. It is easy to see in the bottom chart of Figure 4-8 that this order is maintained. When using one input for each map feature however (top chart), the speeds for the different features cross each other several times, and the end result (red vertical line) predicts that *RoughOpenScatteredTrees* is slightly slower than *EasyRunning*. This is absolutely plausible, as *RoughOpenScatteredTrees* in the area where the training data was obtained is typically a deforested area with leftovers in the form of branches and twigs littering the ground and making fast running difficult. Using the grouped map features, the ANN could not pick up on this. Additionally, with grouped map features the speeds span a smaller range and are quite evenly spaced. It seems that the ANN has difficulties separating them properly.

For the final experiments in Section 4.4, the input vector with one input per map feature was used.

Table 3: Key feature test set - varying forest, all other features constant

Forest	Water	Undergrowth	Paths	Running Slope	Terrain Slope
VeryDifficultToRun	-	-	-	0	0
DifficultToRun	-	-	-	0	0
SlowRunning	-	-	-	0	0
EasyRunning	-	-	-	0	0
RoughOpenScatteredTrees	-	-	-	0	0
RoughOpen	-	-	-	0	0
Open	-	-	-	0	0



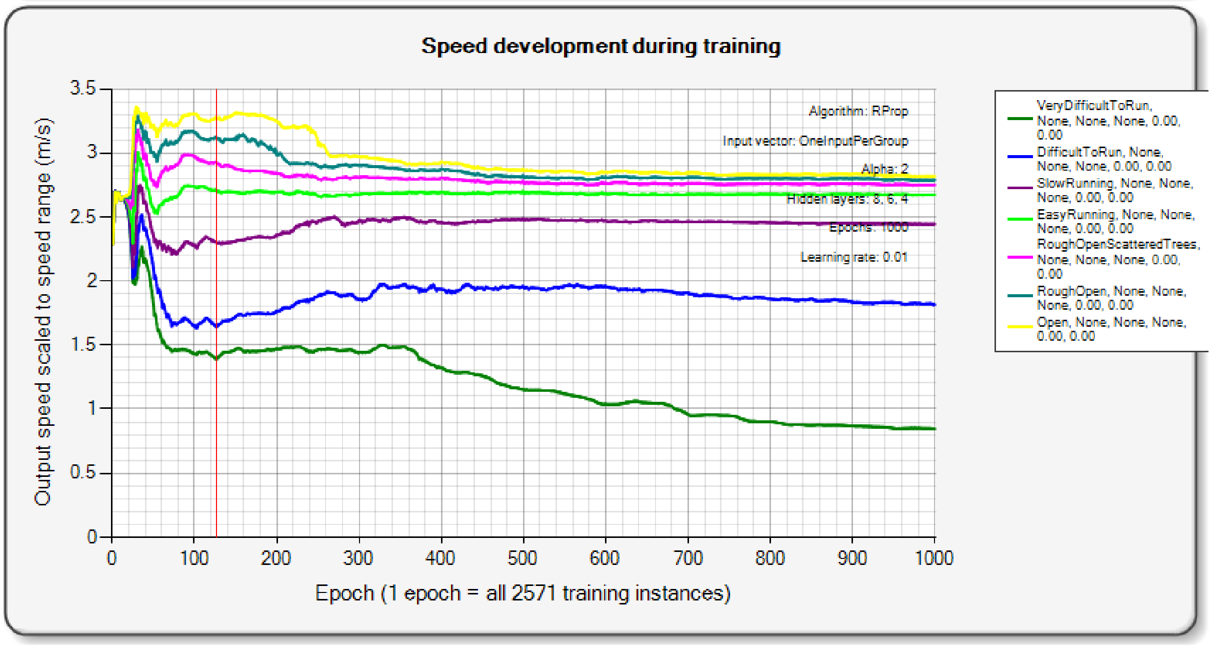


Figure 4-8: Speed plots using the two choices of input vector. Using one input per map feature, the output speeds for the different key feature tests are allowed to develop independently of each other. Notice how the plots cross each other several times. Using one input per map feature group, the output speeds of the different key feature tests seem to be far more dependent on each other, and they never cross. They maintain the same order throughout, which corresponds to ordering them according to input value. (*VeryDifficultToRun* has an input value of 0. *Open* has an input value of 1. The others are in between, in the same order as output here.)

4.3.3 Use of K-fold Cross-validation

When training several ANNs in a row on the same training set and applying them to compute the speeds for a given key feature test set, significant variation in the output speeds was observed. The problem is illustrated in Figure 4-9**Error! Reference source not found.**. Three ANNs were trained with identical learning parameters on almost the same training data, the difference being only that 20 % of the training data, drawn at random, was used for validation in each case to let the learning algorithm know when to terminate training. Some of the variation can probably be attributed to this difference, but there was also considerable variation when the exact same training set was used for each ANN (not shown here). Some variation is to be expected, due to the fact that each weight of each ANN is initialized to a random value (in this case in the range $(-0.5, 0.5)$), which causes them to converge to at least slightly different values. However, given a sufficient amount of training examples representative of the target function, as well as a sufficient number of epochs to converge, the trained ANNs should theoretically be able to produce the same speed predictions with an arbitrarily small margin of error (see Section 2.3.1). The fact that they do not in this case can likely be attributed to noise in the training data; a certain amount of the training examples are not representative of the target function. The more noise the training data contains, the more training examples are needed for the different ANNs to approach a consensus. The different initial conditions allow the ANNs to deal with the noisy data differently from each other, leading them to converge to different local minima of the error space.

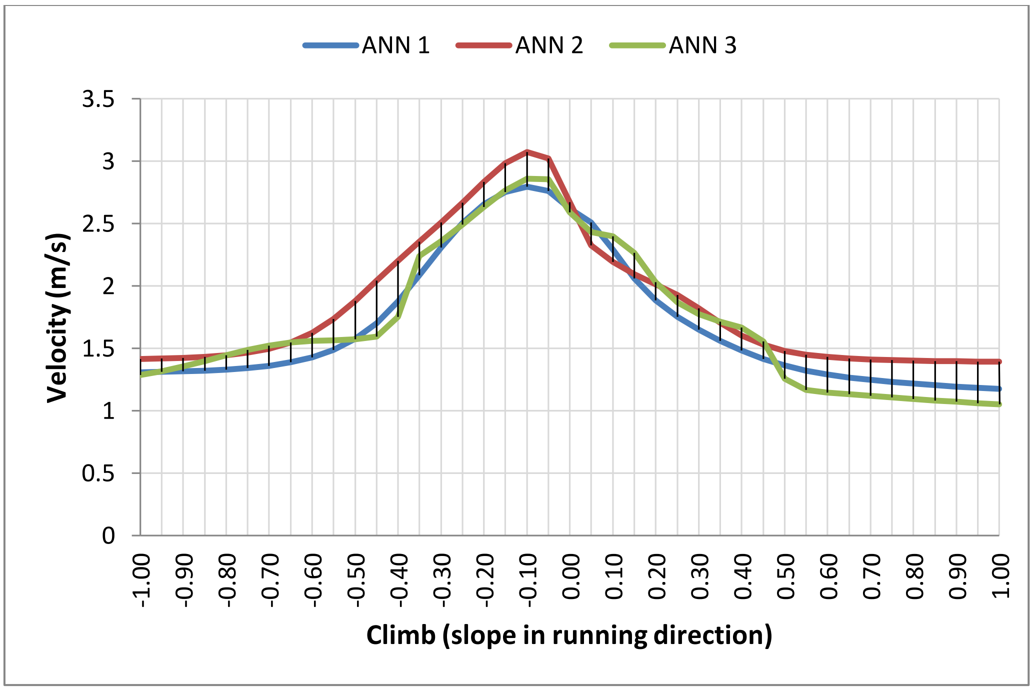


Figure 4-9: Output key feature speeds for three ANNs trained with the same parameters. The same training set was used for each ANN, but 20 % of the training examples were drawn at random to use for validation. The key feature test set used here consists of 41 key feature tests that only vary in running slope. Terrain slope for this test set is the absolute value of the running slope, meaning that the runner is going straight up or down the hill. Forest is EasyRunning. No other map features are present. The black vertical lines indicate the largest difference between two speed predictions for each key feature test case.

In order to alleviate this problem, the use of k-fold cross-validation (described in Section 2.3.2) was attempted. A k-value of 5 was used, so that five networks were trained, each time using a different 1/5 of the training data for validation and termination check. This was done three times, resulting in three committees of 5 ANNs each. The committees were then applied to compute the speeds for the same key feature test set as that of Figure 4-9. The output of the committee was computed by simple averaging of the five member ANNs' outputs. The resulting speeds, shown in Figure 4-10, strongly indicate that a committee of ANNs is more reliable in its speed predictions than a single ANN. Thus, this technique was used for the final experiments in Section 4.4.

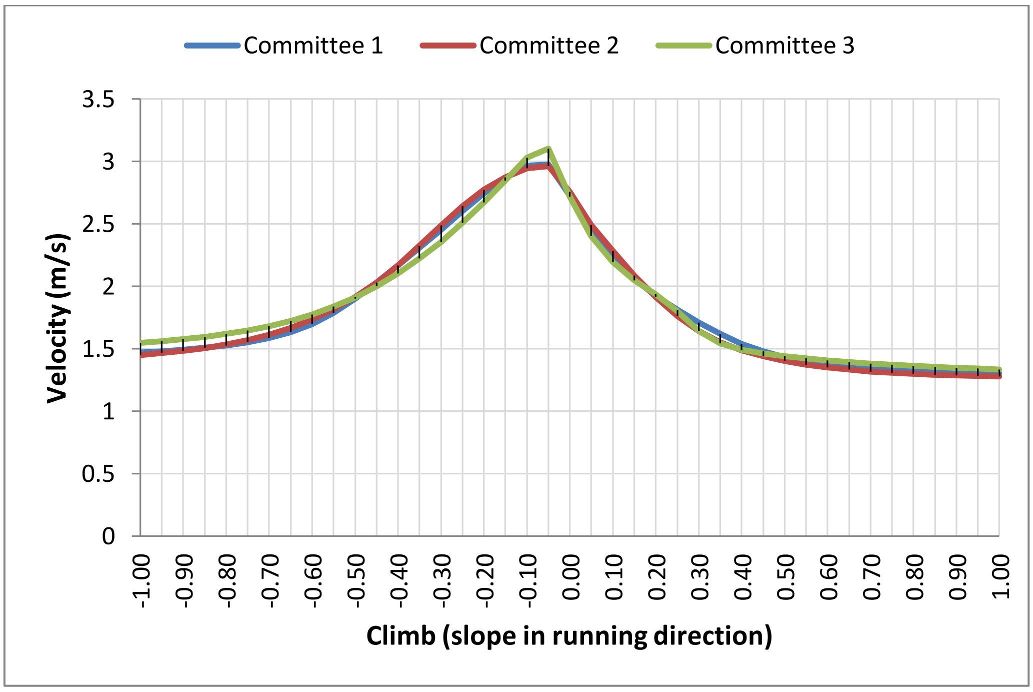


Figure 4-10: Output key feature speeds for three ANN committees trained with the same parameters through 5-fold Cross-validation. When comparing this chart with that of Figure 15, it seems quite evident that a committee of ANNs is more reliable than a single ANN.

4.3.4 Other parameters

Initial experimentation with various settings for the learning parameters indicated a learning rate of 0.01 was appropriate. The alpha value of the Sigmoid function was set to 2. Many different network sizes were tried, and most of them seemed to work equally well. As expected, at least one hidden layer was necessary for the ANN to converge to a plausibly correct state. The effect of using no hidden layers can be seen in Figure 4-11. Based on the experiments, three hidden layers were finally selected, with 8, 6 and 4 hidden neurons for the first, second and third layer. Larger ANNs took longer to train without any obvious increase in accuracy over the validation set. When training on training data with a GPS sampling rate of 1 second, 1000 epochs seemed to be sufficient to guarantee that the lowest validation error was reached for the selected learning parameters. (Usually this was reached within the 100-200 epochs.) It is possible that further optimization of the learning parameters and network size could have led to slightly improved results.

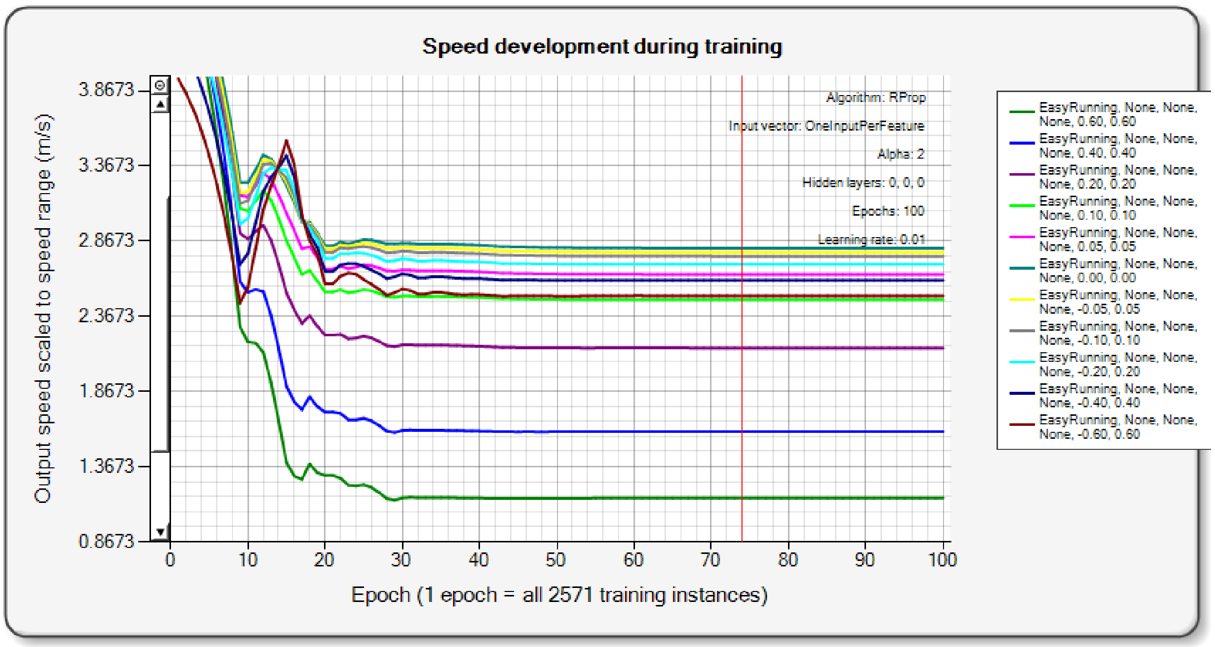


Figure 4-11: Output speeds after each training epoch for the key feature test set of Table 2 as computed by ANN with no hidden layers. This ANN quickly converges to a steady state, but the ANN has failed to learn that large negative values of running slope is almost as bad for running speed as large positive values of running slope. In the legend, running slope is the second last parameter for each test case. Notice that large positive running slopes lead to significantly lower speed. The same cannot be said for large negative running slopes.

4.4 Validation of Model

The implemented system was tested with data from the long distance final of the 2010 World Orienteering Championships in Trondheim. The race was held at Granåsen Ski Arena. A digital version of the orienteering map of the area was obtained, as well as DEMs in various resolutions. GPS tracks of orienteers from the long distance were used for conducting experiments. In addition, a test subject collected GPS tracks by running in the terrain using a GPS logger with higher sampling frequency than that of the GPS data from the competitions. Experiments were conducted with these data as well.

4.4.1 Experiment Outline

A number of experiments were conducted to test the performance of the implemented system. The goal was to determine whether the implemented system can be useful for orienteers in modeling the terrain-speed relationship. Some initial experimentation was done with GPS tracks from orienteers who ran the WOC2010. Unfortunately, with a sampling rate of only one track point per seven seconds, the amount of training data obtainable was apparently too low to obtain any valuable results. For this reason, the experiments described in this section were conducted with GPS tracks with a much higher sampling rate of one track point per second, collected by a test subject. The GPS tracks were collected through two running sessions. For each session, the subject was given a map with an orienteering course where the purpose was to sample as many different terrain feature combinations as possible, rather than to provide interesting or difficult orienteering problems. The controls were placed in easy locations, so that the risk of mistakes and hesitation was very low. The test subject reported after the sessions that no notable mistakes were made. In addition, a third running session was done at a later time, this time to collect data for testing the ANN-based speed model in LeastCostPath. The GPS data was logged with a Garmin Forerunner 910XT sports watch, which has a sampling rate of one track point per second. Before start, the watch was left idle in an open location for a few minutes, so that it could get a proper satellite fix.

The results of the experiments using the ANN committee trained on these data are presented and discussed in the following subsections. The experiment procedure is given here:

1. A set of training examples was created from GPS data, orienteering map and DEM. 20 % of the training examples were reserved for the test set, while the remaining 80 % were used as the training set.
2. An ANN committee was trained on the training set, using 5-fold cross-validation. Thus five ANNs were trained, each time using a different 20 % of the training set for validation and the remaining 80 % for training.
3. The error over the training set and test set for the trained committee were computed, and the distributions of target and output speeds for both sets were compared.
4. Various key feature test sets were presented to the trained committee, and the running speeds computed. Several diagrams were made to visualize the results. The key feature test sets are:
    a. All variants of *Forest* for varying running slope (from -1.0 to 1.0)
    b. All variants of *Paths* for varying running slope (from -1.0 to 1.0)
    c. All variants of *Paths* in combination with all *Forest* types. (The purpose of this is to determine whether *Forest* type affects speed on paths, which it should not.)
    d. Zero running slope, but varying terrain slope. Running in steep terrain should be slow even when you are not ascending or descending.
5. The trained ANN committee was used as a speed model in the modified LeastCostPath application, which was then used to attempt to answer the following questions:
    a. How does the time computed by the LeastCostPath application for a prescribed route compare with the time taken by the orienteer who actually ran that route?
    b. Does the best computed route for a leg match the best route in practice?
6. Point 5 was repeated using the standard speed model of Eq. (2.2) and (2.3), in order to compare the mathematical model with the trained model.

4.4.2 Speed Distribution

Table 4 shows statistical data about target and output speeds for the training set and the test, as well as the mean absolute error (MAE) of the output over both sets after training. Note that the MAE is given both in actual speed (m/s) and relative to the theoretical maximum speed, i.e. the scaling factor of the trained network, which is three times the average speed of the training set (see Section 3.5). Figure 4-12 shows the distribution of target speeds of the training set and test set, as well as output speeds for the same sets, as computed by the trained ANN committee.

Table 4: Target, output and error statistics for Experiment 1.

	Training set	Test set
Number of examples	2571	643
Average target speed	2,65 m/s (6:17 min/km)	2.62 m/s (6:20 min/km)
Max. target speed	6.96 m/s (2:23 min/km)	6.38 m/s (2:36 min/km)
Min. target speed	0.10 m/s (170:53 min/km)	0.12 m/s (142:21 min/km)
Standard deviation target	1.05 m/s	0.99 m/s
Average output speed	2.64 m/s (6:17 min/km)	2.65 m/s (6:17 min/km)
Max output speed	4.82 m/s (3:27 min/km)	4.81 m/s (3:27 min/km)
Min output speed	1.23 m/s (13:31 min/km)	1.23 m/s (13:30 min/km)
Standard deviation output	0.76 m/s	0.74 m/s
MAE	0.5209 m/s	0.5174 m/s
MAE relative to theoretical max. speed (3 * average speed)	0.0656	0.0651

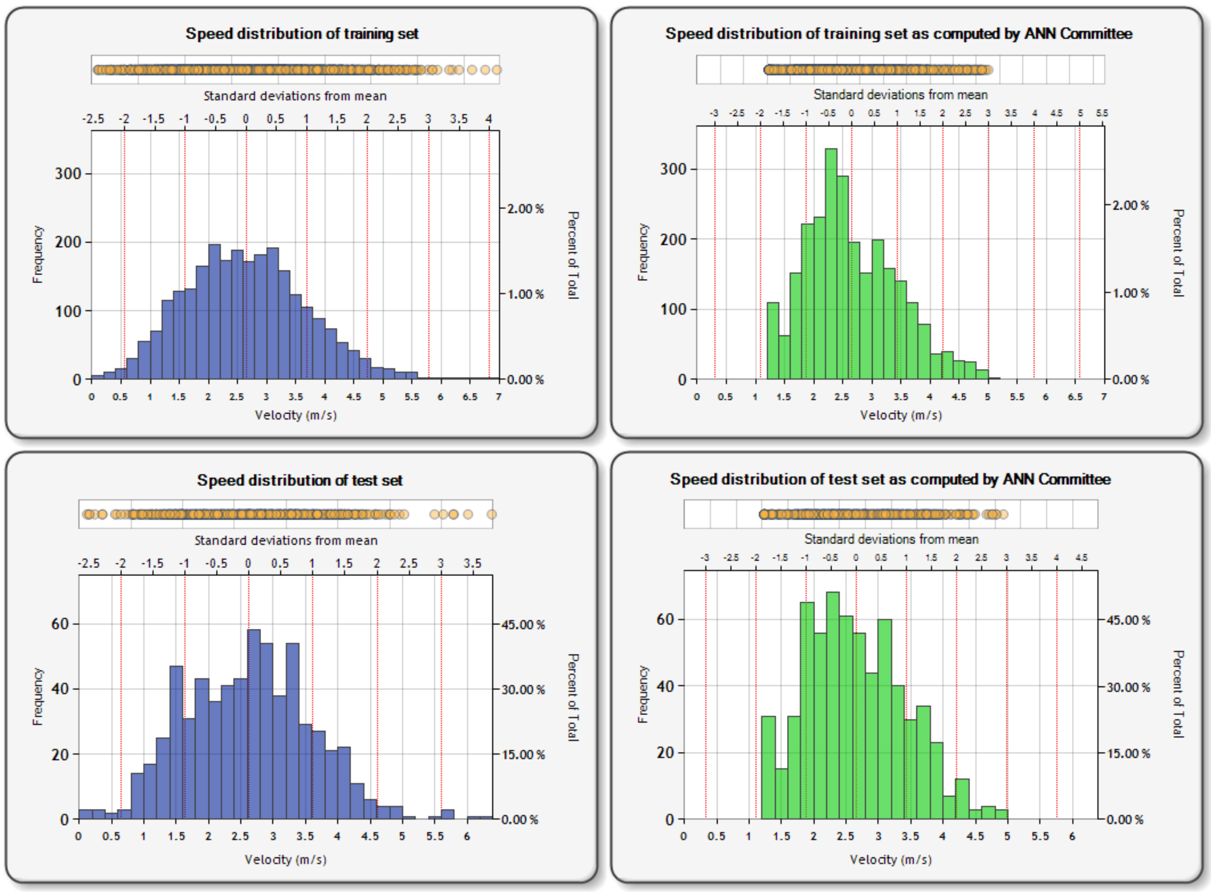


Figure 4-12: Speed histograms for output and target speeds for the training set and the test set.

The distribution of the target speeds appears to be approximately normal. The distribution of the test set's target speeds is slightly more irregular, due to the smaller size of the set. Looking at the output speed distribution (of both training set and test set), it seems that the lowest speeds (below 1.2 m/s) have been bumped up to higher speeds, causing a squashing of the output distribution below the mean, and resulting in the peak of the distribution being placed slightly below the mean. This can also be observed as a lower standard deviation of speeds than in the target distribution. Notice that the number of examples occurring in the bin around the mean is approximately the same in the target distribution and the output distribution, and that the average output speed is almost identical to the average target speed. The likely reason for the squashing below the mean will become apparent in the first key feature test. See below.

4.4.3 Key Feature Test A: Forest Types and Varying Running Slope

Figure 4-13 shows how the output speed varies with the running slope in different types of forest. The chart reveals some interesting differences between the various forest types. For *RoughOpen*, *EasyRunning*, *RoughOpenScatteredTrees* and *SlowRunning*, the fastest speed is found at a running slope of -0.05. Both increasing and decreasing the running slope from this optimum lowers the speed, which is not surprising; increasing the slope makes running more physically exhausting, and decreasing it makes running more technically challenging, due to the uneven nature of the terrain surface. For the forest type *Open*, speed is notably higher when running downhill. This is also natural, as the map symbol *Open* typically means an open grass field with a quite even surface, making downhill running very easy. *VeryDifficultToRun* maintains a fixed value independent of the running slope. Even though this is not implausible, little should be concluded from this chart about the precise speed in that forest type, because it is nearly non-existent in the training set. This naturally makes it very difficult for the ANN to learn the correct speed.

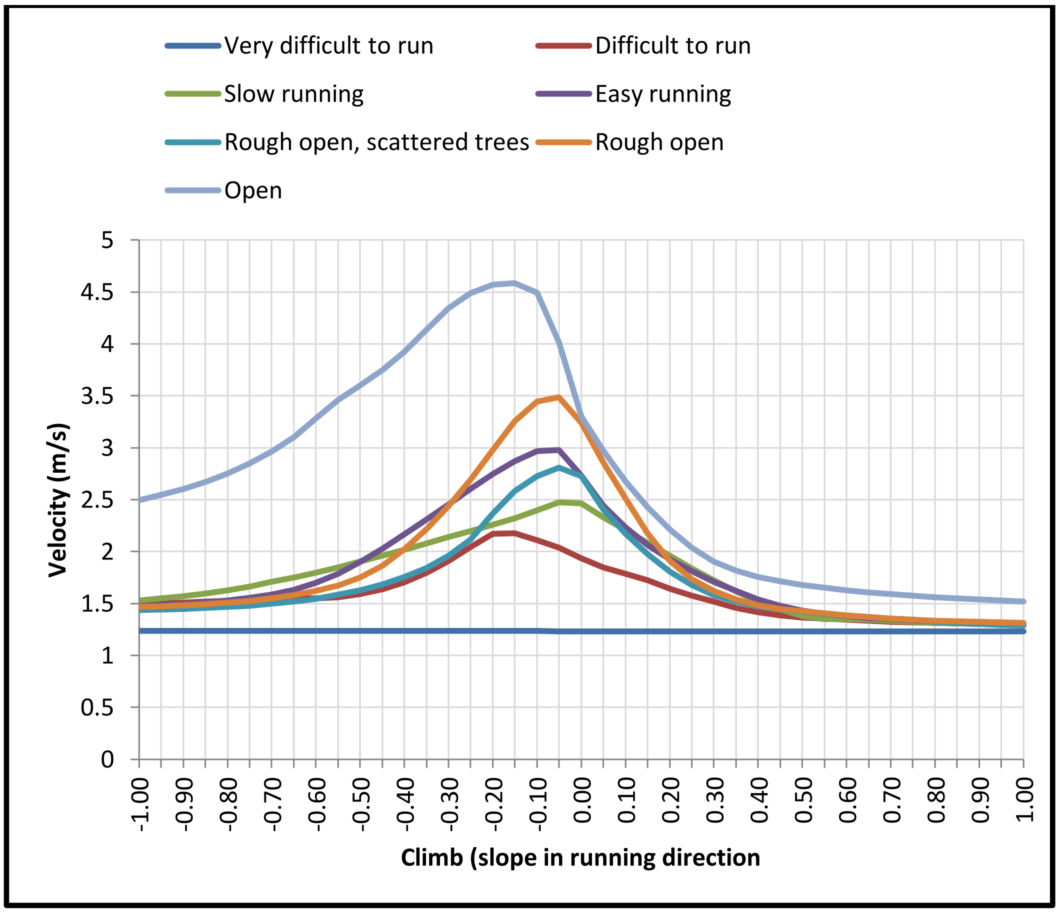


Figure 4-13: Output speed as a function of running slope and forest type.

There are some surprises as well. *DifficultToRun* being at its fastest at a running slope of -0.2 is certainly one, especially since the speeds for *SlowRunning* and *EasyRunning* start diminishing beyond -0.05. This may be attributed to the relatively modest size of the training set, and that the denser forest types are less abundant. Similarly, output speed at extreme running slopes should not be trusted, as the amount of representative training data is limited.

One thing that can clearly be seen from the figure is that the running speeds for all forest types seem to converge towards approximately the same constant value as the running slope approaches extreme values, both in the positive and negative direction. Observe for instance that the speed difference between a slope of 0.5 (27°) and 1.0 (45°) is extremely small. This is highly unlikely to be the case in reality. For running slopes beyond 1.0, there is practically no difference at all (not visible in the figure). This can probably be attributed to the way running slope is encoded in the input vector (see the discussion on normalization in Section 3.4). And here is the most likely reason for the squashed output speed distribution of Figure 4-12. The lowest speeds in the target distribution were probably those of training examples with extreme running slopes. In retrospect, a different representation should have been chosen to keep the running slope in the $(0, 1)$ range while being able to represent all slopes. This could for instance have been done by scaling the degree range of $(-90°, 90°)$ into the $(0, 1)$ range.

The fact that some of the speed predictions are less certain than others is illustrated in Figure 4-14. The chart shows three of the speed curves from Figure 4-13, with added error bars that display the mean average deviation of the outputs of the five ANNs of the committee. A large mean average deviation indicates that the ANNs disagree, while a small deviation indicates agreement. When there is little training data available for particular combination of input features, it is to be expected that the different ANNs will close the gaps differently. This seems to be the case for extreme running slopes. Note that *Open* has a particularly high deviation for large negative running slopes, which coincides with the fact that the training data consists of only a very short stretch of downhill running in such an open area. (Uphill in open area happens to be more abundant in the training set.)

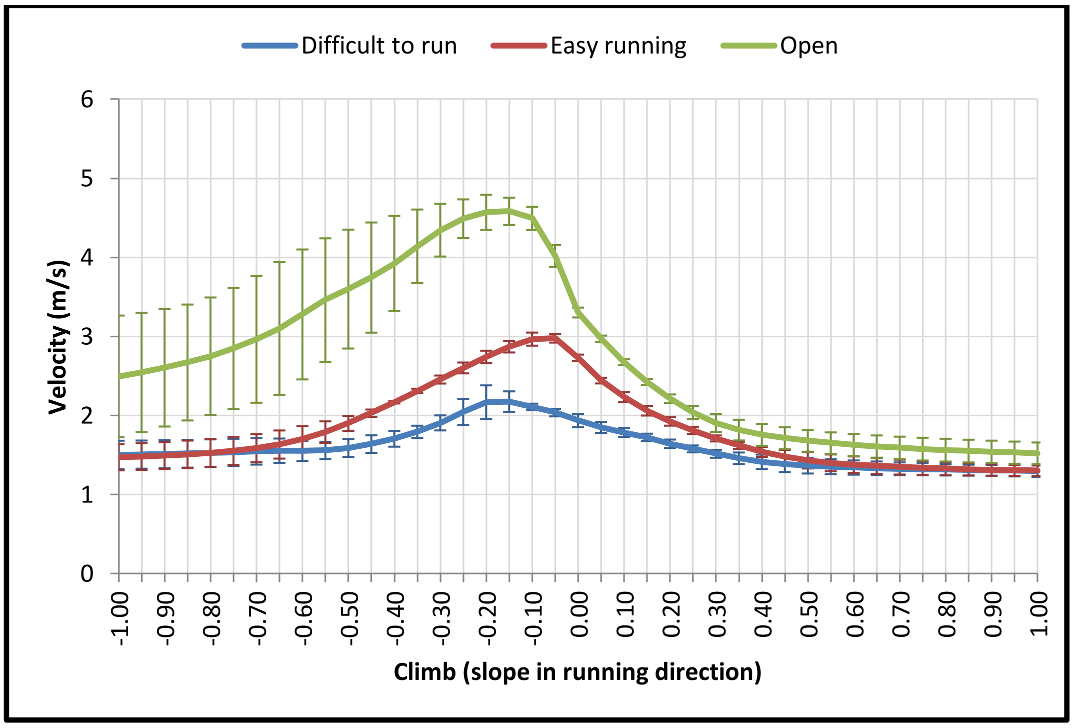


Figure 4-14: Three of the output speed curves in Figure 4-13, with error bars showing the mean absolute deviation of the outputs of the five ANNs in the committee.

4.4.4 Key Feature Test B: Paths and Varying Running Slope

Figure 4-15 shows output speed as a function of running slope and type of path. There are considerably more problems here than in Figure 4-13, which can almost certainly be attributed to lack of relevant training data. Downhill *Road* is rightly fast, but there is really no reason why *NarrowRide* should be faster than *Road* uphill. Also, *FootPath*, which is a quite large path, should be at least as fast as *SmallPath*. The reason it is not is probably because there is almost no training data for it, while *SmallPath* is quite well represented. It should be noted that *SmallPaths* in the test area were highly runnable and probably not inferior to *Footpath*. In retrospect, those two could perhaps have been combined into one ANN input, similar to what was done with *Road* and *VehicleTrack*.

When looking at the mean absolute deviations of *Road*, *Footpath* and no path in Figure 4-16, it is easy to see that the ANNs of the committee are much more in agreement when there is no path. (The no path curve corresponds to the *EasyRunning* curve in Figure 4-13.) This is a strong indication that more training data is needed to produce an accurate model of running speed on paths and roads. In addition, this is probably a case where using the other input vector type might render better results, since grouping all forest features together in one input neuron would likely entail at least some degree of similar treatment for all of them.

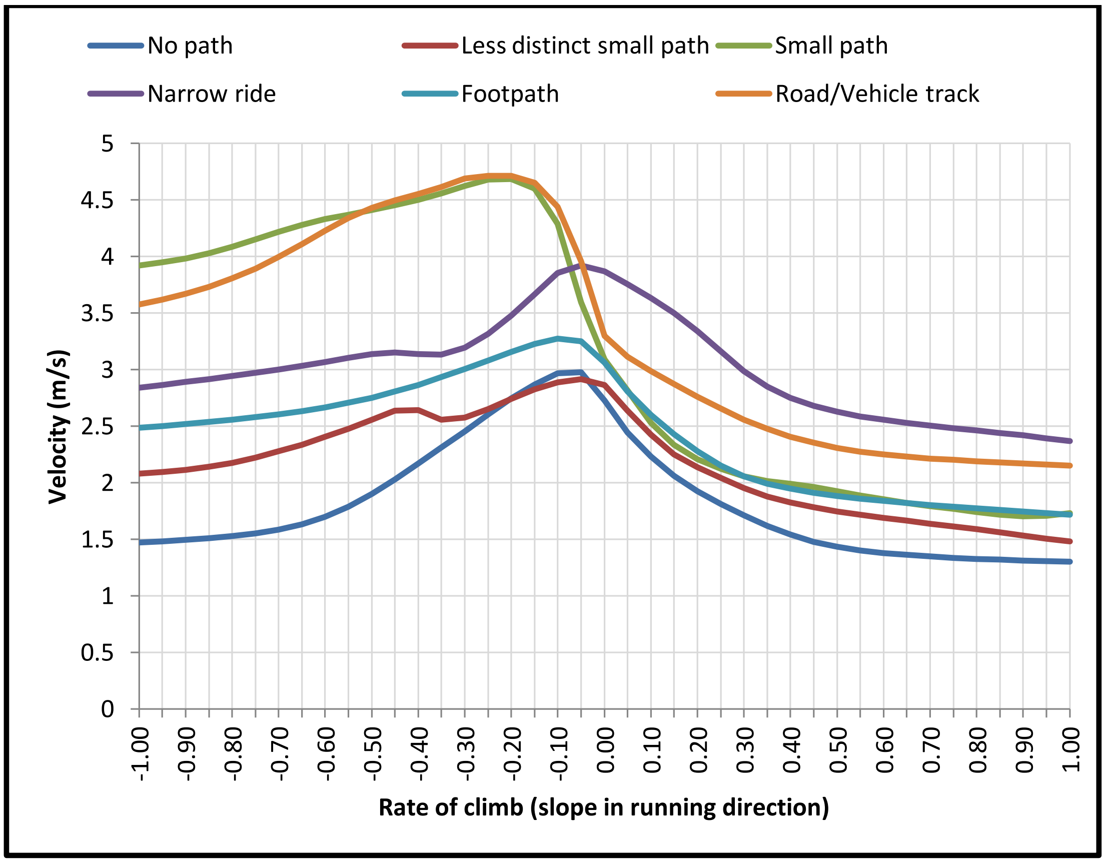


Figure 4-15: Output speed as a function of running slope and path type.

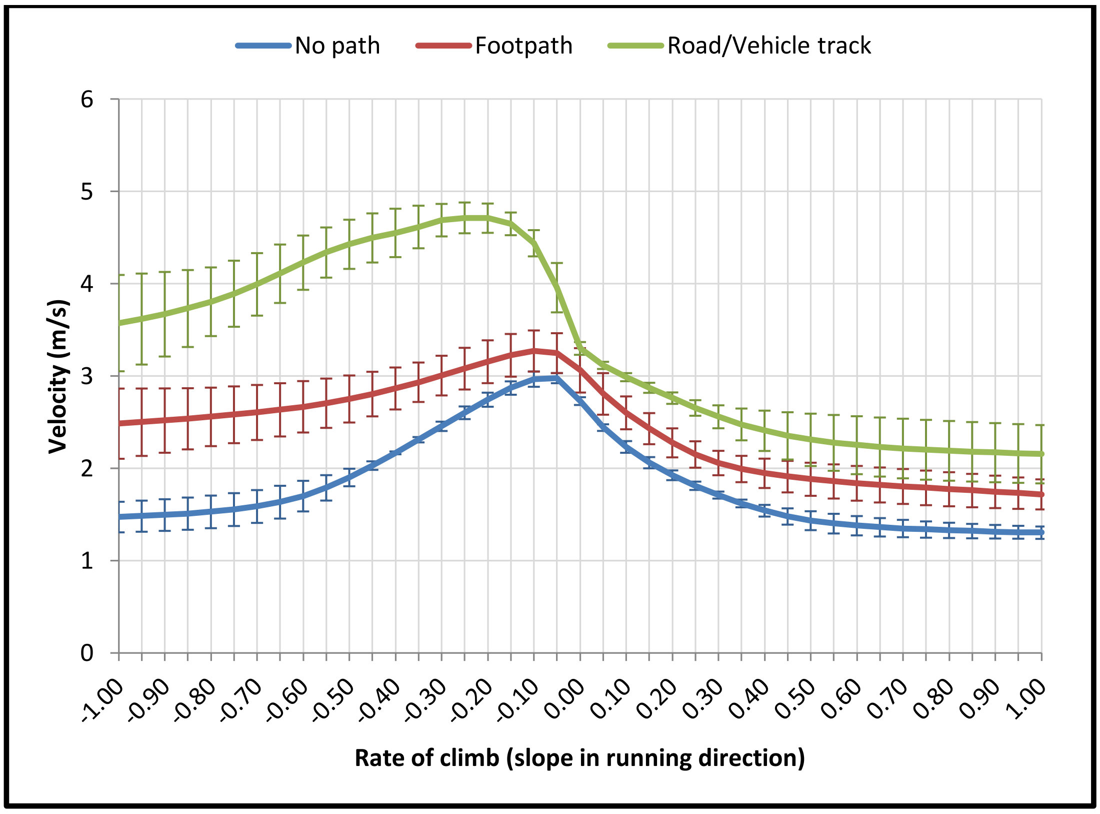


Figure 4-16: Three of the output speed curves in Figure 4-15 with error bars showing the mean absolute deviation of the outputs of the five ANNs in the committee.

4.4.5 Key Feature Test C: Paths and Forest Types

When running on a path or a road, the speed should not be dependent on the type of forest that the path/road runs through. This test will reveal whether the ANN committee has learned this independence.

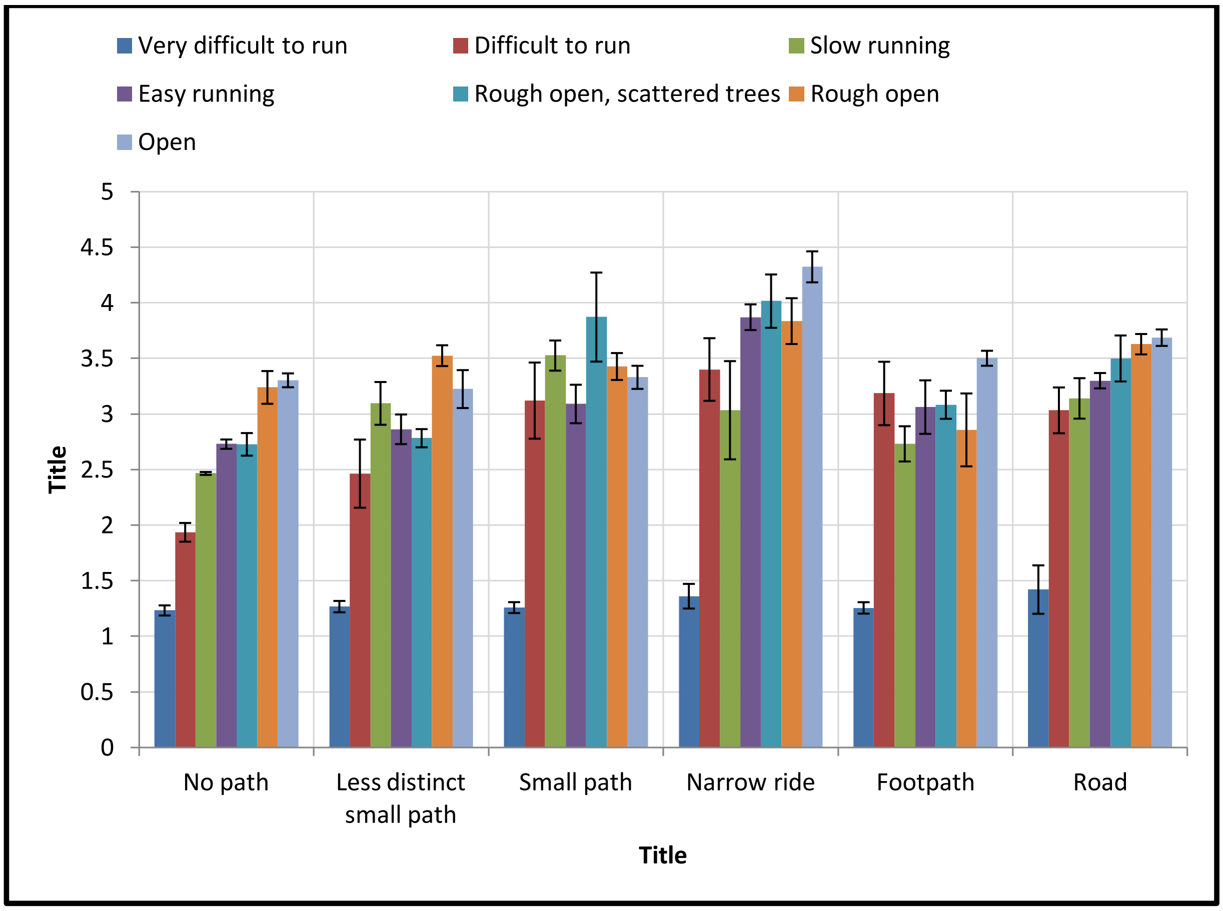


Figure 4-17: Speeds for different types of path in relation to the forest type which the path runs through.

Figure 4-17 shows the output speeds for all path types in relation to the forest type which the path runs through. Ideally all the columns of each cluster in the chart, except for the first one, should be of equal height. *VeryDifficultToRun* is almost non-existent in the training data, which explains why the ANNs have not been able to learn that paths and roads through such areas are just as good as paths and roads through other types of forest. If we ignore *VeryDifficultToRun*, we can see that the variation in running speed for the different forest types is less with the presence of a path/road than without, which gives reason for hope. Given sufficient training data, the ANN committee would probably be able to learn the independence better.

A way to alleviate this problem would be to overrule the Forest inputs to *EasyRunning* upon the presence of a path, both during training and subsequent query of the ANNs. This would effectively deny the ANNs the possibility of differentiating between for instance a road through a dense forest and a road through an open field, since both would appear to the ANN as a road through regular forest.

4.4.6 Key Feature Test D: Zero Running Slope, Varying Terrain Slope

Running in steep terrain is slow, no matter if you are going up, down or staying at the same altitude, running along the hillside. An exception to this would be if you are running on a path/road along the hillside. Often, the ground will be level along the path, forming a kind of shelf with the terrain climbing up on one side and going down on the other. Depending on the resolution and detail of the DEM that is used to compute slope information, this shelf may or may not be present in the training data. Furthermore, if the GPS track is off by a few meters, the path correction technique discussed in Section 3.3.2 will allow the track point to be classified as on a path, but it will not correct the erroneous slope information. In such cases, the ANN should learn that the terrain slope matters little when a path is present.

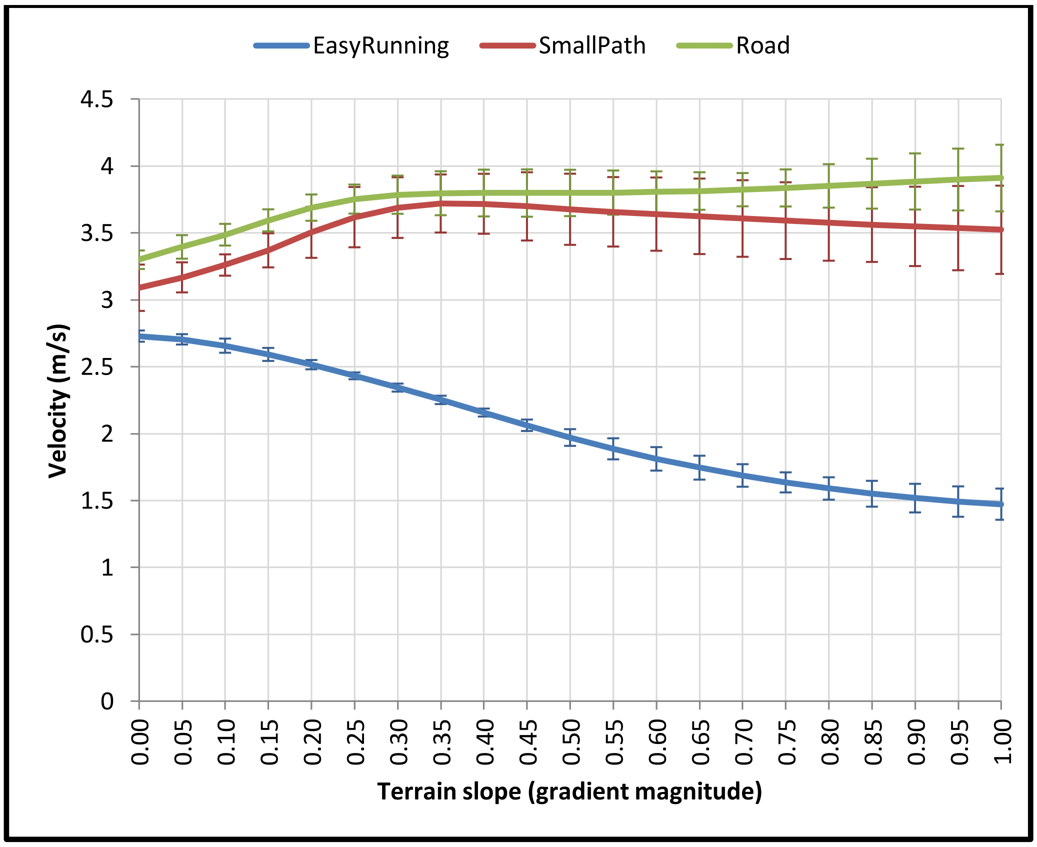


Figure 4-18: Output speed as a function of terrain slope and running surface. Running slope is zero throughout. Road/path typically implies that the ground is level even if the terrain slope is steep. This is not the case for regular forest ground (*EasyRunning*).

The result of the terrain slope test can be seen in Figure 4-18. While speed decreases with increasing terrain slope in regular forest, it is fairly stable on path/road for most of the slope range , although it actually increases as the slope creeps from 0 to 0.3. The reason for this is unclear, although it is not unlikely that lack of relevant training examples is the culprit once more.

Note that running speed in *EasyRunning* seems to converge to a constant value as the terrain slope increases beyond 1. This is likely too soon, and it indicates that the terrain slope input representation, like the running slope input representation, should be replaced in the future.

4.4.7 Route Choice Test in LeastCostPath

Figure 4-19 shows a route choice leg (from control A to control B) with four alternative routes specified. The test subject ran each of the four routes (the routes are GPS tracks). The time spent for each route is indicated in the figure. Now we want to find out whether the required time for each route according to the trained ANN committee matches the observed times.

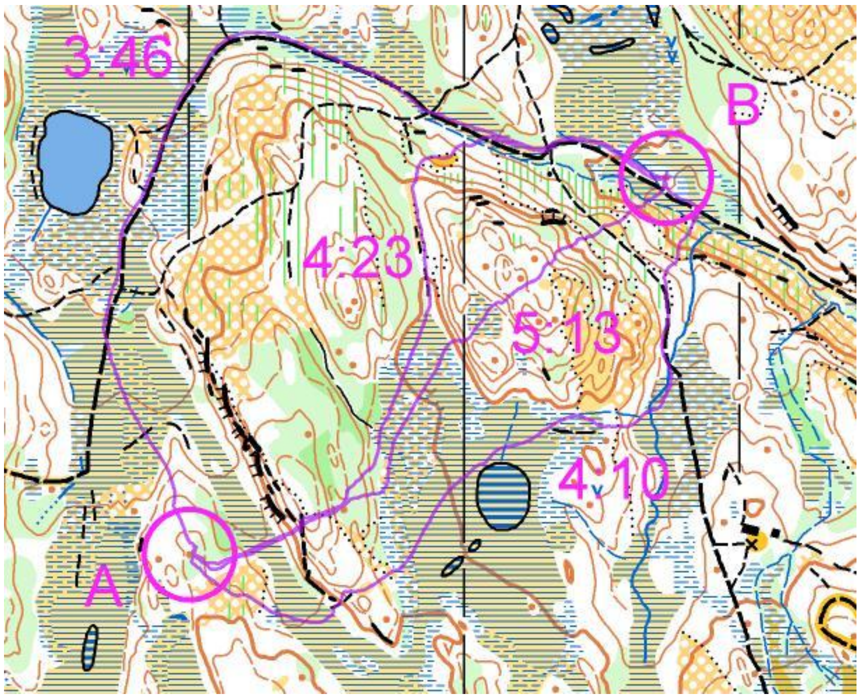


Figure 4-19: Route choice test. The four routes (GPS tracks) on the map were all run by the test subject. The time spent for each route is indicated next to the route. The left route on the footpath is the clear winner.

The LeastCostPath program was used to compute the time cost for each route (using the corridor method described in Section 3.8 with a corridor width of 10 meters), and to compute the time model and route model for the leg, as explained in Section 2.4.2. The grid size of the search grid was 1m x 1m. The results of the time cost computations for each route are given in Table 5. The time model and route model are shown in Figure 4-20.

Table 5: Results of the route choice test using the ANN Committee Speed Predictor

Route	**Real time cost**	**Computed time cost**	**Error**	
Left	3:46	4:47	+1:01	+27.0 %
Straight-left	4:23	4:13	-0:10	-3.8 %
Straight	5:13	3:56	-1:17	-24.6 %
Right	4:10	4:14	+0:04	+1.6 %
All combined	17:32	17:10	-0:22	-2.1 %

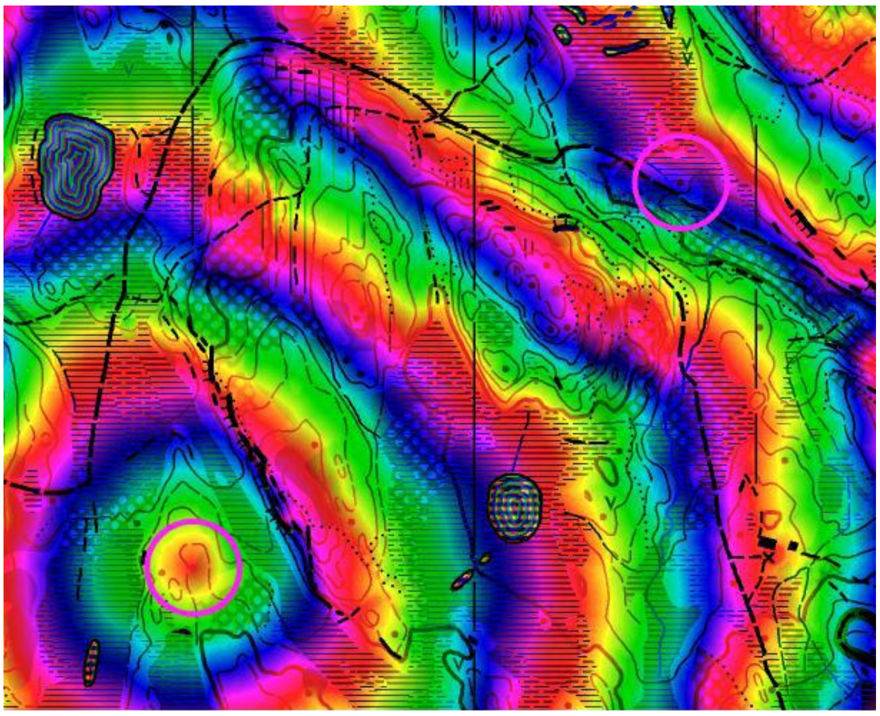
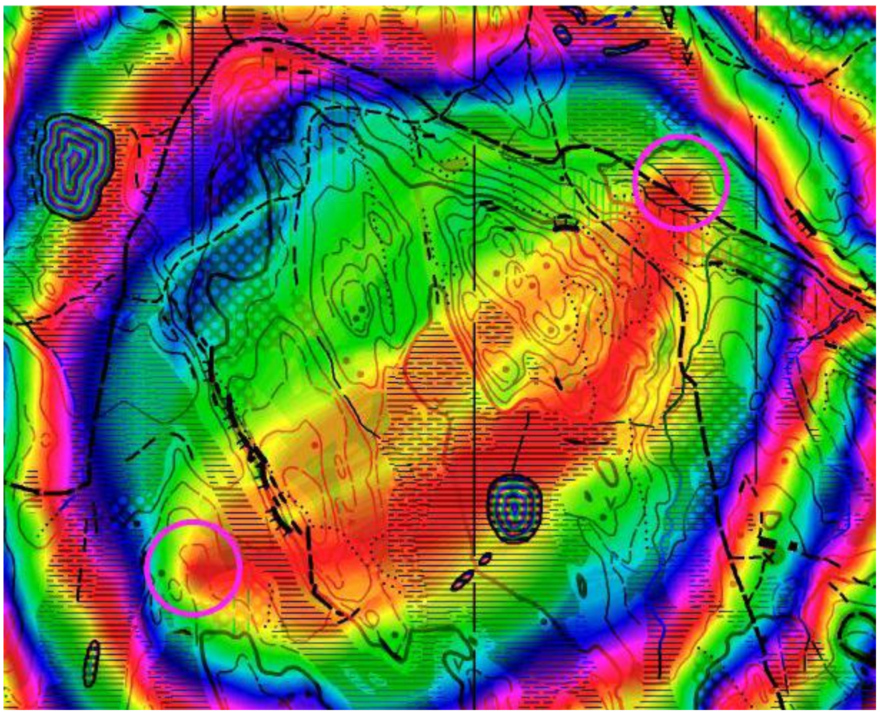



Figure 4-20: Time model (top) and route model (bottom) for the route choice leg of Figure 4-19, using ANN Committee Speed Predictor. In the time model, one transition from red to red equals one minute of time. A transition from one color band to the next (for instance red to yellow) equals 10 seconds. In the route model, one transition from red to red equals a time loss of one minute compared to the optimal route. A transition from one color band to the next (in increasing order, see Section 2.4.2) equals a time loss of 10 seconds, compared to the optimal route. Note that forbidden areas (the small lake and the dangerous swamp) are implemented simply by enforcing a very low running speed, which causes the densely packed rings of color that can be seen in the figure.

The results are not too surprising, given what we have seen in the key feature tests. The ANN committee speed model computes the best time of 3:56 for the straight route. In reality, this is in fact the worst route, due to the large amount of steep ascent and descent. In fact the descent from the highest point of the route down to the control is the real time stealer. As we have seen, the ANN committee has trouble differentiating between running slopes beyond a certain level, and so the steep hills are not considered as slow as they should be. Thus, the straight route, which is after all the shortest, receives the lowest computed time. The best route in practice is the left route, which consists almost entirely of running on a footpath. As was demonstrated in the key feature tests, the ANN committee's belief of the running speed on that particular path type is far too low, due to the fact that it is almost non-existent in the training data. Consequently, this route does not fair too well in the computed costs. For the right route choice, which runs mostly on flat marshes, the computed cost and the real cost match very well. Flat marshes are well represented in the training data, so this is an indication that the speed predictions are accurate when the ANN committee has been trained with sufficient and relevant data. The computed time for the straight-left route is also quite good, since it avoids the slow descent of the straight route, though the steep climb in the beginning takes more time than computed.

When looking at the computed and real time cost for all the routes combined, the error is actually very low. This is probably a pure coincidence in this case, as there was one route which was approximately as much too slow as another was too fast.

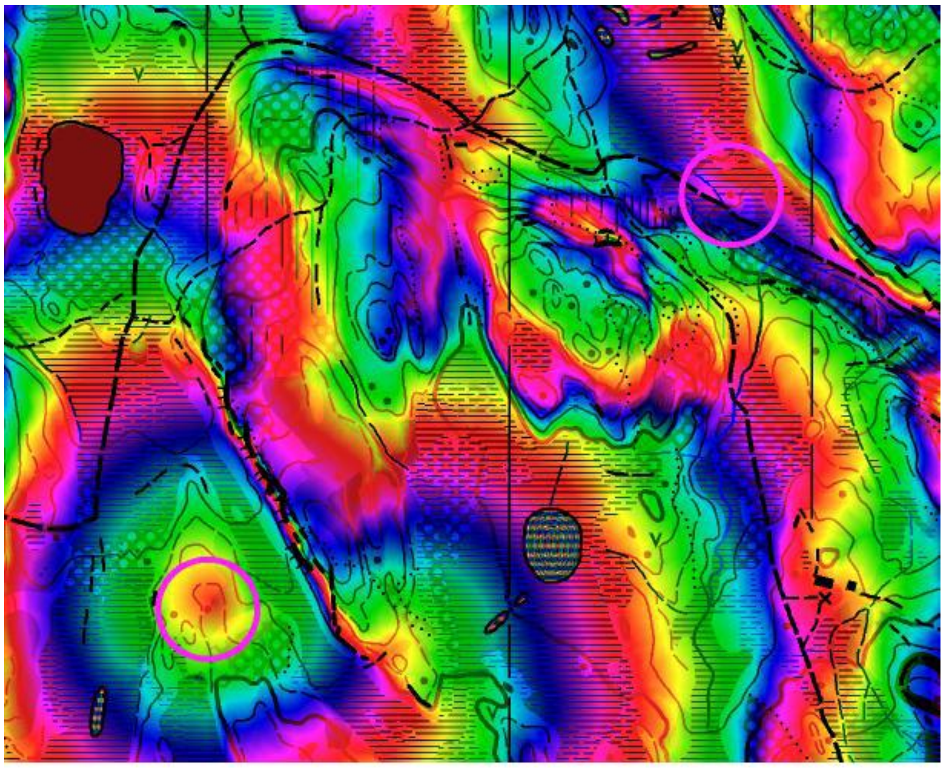
On a side note, the cost of the optimal route computed by LeastCostPath with the ANN committee speed model (the pure red color band in the route model) was 3:43. Even without having actual real data to compare this with, we can safely say that this too is unrealistic, when comparing with the other routes.

4.4.8 Route Choice Test in LeastCostPath with Standard Speed Model

For comparison, the standard speed model included with LeastCostPath, discussed in Section 2.4.1, was also used to compute the time cost for the four route choices, as well as the complete time model and route model. Since that model computes time costs in terms of equivalent distance flat road/path, we need to know the running speed of the test subject on a flat road/path. This was done by selecting all training examples in the entire set of training data that were on a path and had a running slope in the range $(-0.02, 0.02)$. In degrees this equals a range of $(-1.15°, 1.15°)$. A total of 121 such examples were found. Then the average speed of these examples was computed. The result was 3.5220 m/s (equivalently 4:43 min/km). The results of the cost computations with the standard speed model were divided by this number to obtain the actual time in seconds. The results of the cost computations for the four routes are given in Table 6, while the time model and the route model are shown in Figure 4-21.

Table 6: Results of the route choice test using the standard speed model

Route	Real time cost	Computed time cost	Error	
Left	3:46	5:05	+1:19	+35.1%
Straight-left	4:23	5:41	+1:17	+29.7%
Straight	5:13	6:36	+1:23	+26.5%
Right	4:10	5:00	+0:50	+20.2%
All combined	17:32	22:23	+4:50	+27.6%



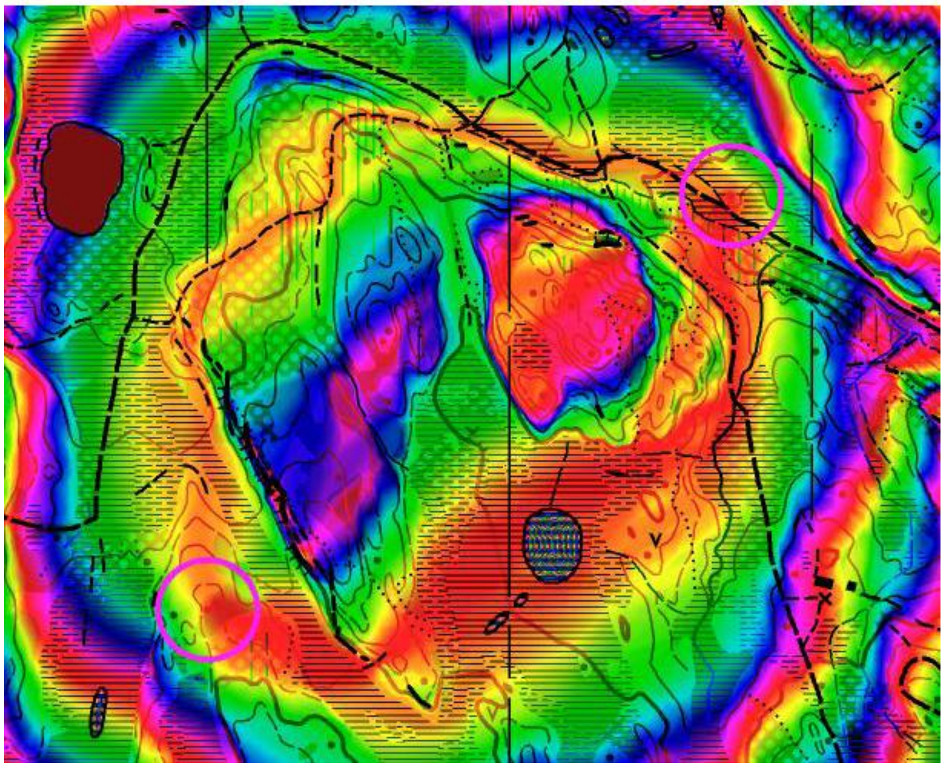


Figure 4-21: Time model (top) and route model (bottom) for the route choice leg of Figure 4-19, using the standard speed model given in Eq. (2.2) and (2.3). Here, one transition from red to red corresponds to the time required to run 256m on a flat road, just like in Figure 2-4.

In terms of accuracy of the absolute costs of the different routes, this speed model does quite poorly, overestimating all costs by a wide margin. (This may be in part because the test subject was not in the best road running shape.) However, the ratios between the different routes are more in sync with reality than those calculated with the ANN Committee as speed model. Still, the computed optimal route here is not the best one that was seen in practice. There is an almost optimal route choice using the small path on the left, but this route is unlikely to be better than taking the larger path (footpath) in reality, as it involves quite a bit of ascent and descent, while the footpath is flat or gentle downhill almost all the way, and only slightly longer. In any case, the routes that can be discerned in this route model are much more in line with what one would expect than those obtained with the ANN committee as speed model, based on the real time costs of the routes.

5 Conclusion and Future Work

The results presented in the previous chapter show that the use of ANN learning to model running speed in orienteering is plausible. In its current state, the implemented system cannot produce speed models that are really useful for an orienteer. The biggest problem is that the trained speed model is not able to differentiate slopes properly. As the running slope slowly approaches infinity, the running speed rather quickly converges to a constant value, which appears to be far too high. The chosen representation for the running slope in the input vector of the ANN is likely to blame, as the range of that particular input is much higher than that of the other inputs of the ANN. Even though it often stays within the $(-1, 1)$ range, $(-\infty, \infty)$ is theoretically possible. This is something the Sigmoid activation function cannot handle. Very high input values to a Sigmoid unit will always produce almost the same outputs, even if the inputs are vastly different from each other. The same applies for low input values. Replacing the running slope input representation, and the terrain slope representation, is thus a priority in future work on this method. An obvious approach would be to use angle of inclination, which is $(-90, 90°)$ for running slope and $(0, 90°)$ for terrain slope, and scale these ranges into the $(0, 1)$ range.

It also seems clear from the experiments that more training data should be used to obtain good results. The route choice test indicated that in areas where the terrain features are such as were well represented in the training set, the learned speed model works well. Some map features were almost non-existent in the training set, which of course makes predicting the speed for those features very difficult.

This thesis has demonstrated that the proposed method is a viable approach to the task of modeling running speed in orienteering, but improvements in various aspects are needed to make it useful as a tool for orienteers.

6 Works Cited

Arnet, F., 2001. *Interpolation von DHM25 Level 2.* Wabern (CH): Bundesamt für Landestopographie.

Arnet, F., 2009. Arithmetical route analysis with examples of the long final courses of the World Orienteering Championships 2003 in Switzerland and 2005 in Japan. *Scientific Journal of Orienteering,* pp. 4-21.

Ciresan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J., 2010. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *Neural Computation,* 22(12).

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *MATHEMATICS OF CONTROL, SIGNALS, AND SYSTEMS (MCSS),* 2(4), pp. 303-314.

ESRI, 2012. *ArcGIS - Mapping and Spatial Analysis for Understanding Our World.* [Online] Available at: http://www.esri.com/software/arcgis/index.html
[Accessed 13 February 2012].

Fahlman, S. E. & Lebiere, C., 1990. *The Cascade-Correlation Learning Architecture,* Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.

Forbord, R. & Bjartnes, A., 2010. *Orienteering map: Jervskogen (WOC2010).* s.l.:s.n.

Forbord, R., Valstad, R. & Treekrem, K., 2010. *Orienteering map: Leinstrandmarka (WOC2010).* s.l.:s.n.

Freislaben, B., 1992. *Stock market prediction with backpropagation networks.* s.l., s.n., pp. 451-460.

Hager, J. W., Behensky, J. F. & Drew, B. W., 1989. *The Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS). DMA Tech. Manual 8358.2,* Fairfax, VA: Defense Mapping Agency.

Hashem, S., 1997. Optimal linear combinations of neural networks. *Neural Networks,* 10(4), pp. 599-614.

Hayes, M. & Norman, J. M., 1984. Dynamic Programming in Orienteering: Route Choice and the Siting of Controls. *The Journal of the Operational Research Society,* 35(9), pp. 791-796.

Hertz, J. A., Palmer, R. G. & Krogh, A. S., 1991. *Introduction to The Teory of Neural Computation.* s.l.:Westview Press.

Hornik, K., Stinchcombe, M. & White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural Networks,* 2(5), pp. 359-366.

Jayalakshmi, T. & Santhakumaran, D. A., 2011. Statistical Normalization and Back Propagation for Classification. *International Journal of Computer Theory and Engineering,* 3(1), pp. 89-93.

Kim, D., 1999. Normalization methods for input and output vectors in backpropagation neural networks. *International Journal of Computer Mathematics,* 71(2), pp. 161-171.

Kirillov, A., 2012. *AForge.NET.* [Online]
Available at: http://www.aforgenet.com/
[Accessed 28 6 2012].

Kocbach, J., 2011. *O-training.net - GPS Analysis for Orienteering: All the Basics!.* [Online]
Available at: http://o-training.net/blog/2011/04/13/gps-analysis-for-orienteering-the-basics/
[Accessed 27 05 2012].

Mitchell, T. M., 1997. *Machine Learning.* Carnegie Mellon University: McGraw-Hill Book Co.

Myrvold, B. O., 1996. Is it Possible to Find a "Best" Route - A Look at Accuracy and Significance in Route Choice Comparison. *Scientific Journal of Orienteering,* 12(1), pp. 19-36.

Naismith, W. W., 1892. Untitled. *Scottish Mountaineering Club Journal,* Volume 2, p. 135.

OCAD, 2012. *OCAD - Smart for Cartography.* [Online]
Available at: http://www.ocad.com
[Accessed 13 February 2012].

Park, J. & Sandberg, I. W., 1991. Universal approximation using radial-basis-function networks. *Neural computation,* 3(2), pp. 246-257.

Persson, B. et al., 2000. *International Specification for Orienteering Maps.* [Online]
Available at: http://orienteering.org/resources/mapping/
[Accessed 5 2012].

Riedmiller, M. & Braun, H., 1993. *A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm.* Inst. fuer Logik, Komplexitat und Deduktionssyteme, Karlsruhe Univ., s.n., pp. 586-591.

Rowley, H. A., Baluja, S. & Kanade, T., 1998. Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(1), pp. 23-38.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J., 1986. Learning representations by back-propagating errors. *Nature,* Volume 323, pp. 533-536.

Sagberg, T., 2012. *Klassifisering av bevegelsesmønster hos orienteringsløpere (Classifying motion types in orienteering), M.Sc. thesis,* Dept. of Mathematical Sciences and Technology, Norwegian University of Life Sciences, Ås: s.n.

Scarf, P., 2007. Route choice in mountain navigation, Naismith's rule, and the equivalence of distance and climb. *Journal of Sports Sciences,* 25(6), pp. 719-726.

Varis, P., 2012. *GPSSeuranta.* [Online]
Available at: http://gpsseuranta.net/
[Accessed 2 2012].

Weltzien, E., 1979. *Studie av elite-o-løperes veivalg,* Oslo: Norges Orienteringsforbund.

Weltzien, E., 1983. *Veivalg.* Oslo: Norges Orienteringsforbund.

WOC2010, 2010. *WOC2010.* [Online]
Available at: http://www.woc2010.com/
[Accessed 10 02 2012].