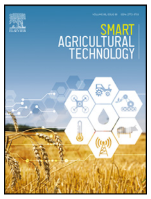
Smart Agricultural Technology 1 (2021) 100017

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.elsevier.com/locate/atech



Multimodal performers for genomic selection and crop yield prediction

Håkon Måløy[a,*], Susanne Windju[b], Stein Bergersen[b], Muath Alsheikh[b,c], Keith L. Downing[a]

[a] Department of Computer Science, NTNU, Sem Sælandsvei 9, Trondheim, Norway
[b] Graminor, Bjørke Gård, Hommelstadvegen 60, Ridabu, Norway
[c] Department of Plant Sciences, Norwegian University of Life Sciences, P.O. Box 5003, 1432 ÅS, Norway

A R T I C L E I N F O

Keywords:
Genomic selection
Yield prediction
Deep learning
Attention models
Barley

A B S T R A C T

Working towards optimal crop yields is a crucial step towards securing a stable food supply for the world. To this end, approaches to model and predict crop yields can help speed up research and reduce costs. However, crop yield prediction is very challenging due to the dependencies on factors such as genotype and environmental factors. In this paper we introduce a performer-based deep learning framework for crop yield prediction using single nucleotide polymorphisms and weather data. We compare the proposed models with traditional Bayesian-based methods and traditional neural network architectures on the task of predicting barley yields across 8 different locations in Norway for the years 2017 and 2018. We show that the performer-based models significantly outperform the traditional approaches, achieving an R^2 score of 0.820 and a root mean squared error of 69.05, compared to 0.807 and 71.63, and 0.076 and 149.78 for the best traditional neural network and traditional Bayesian approach respectively. Furthermore, we show that visualizing the self-attention maps of a Multimodal Performer network indicates that the model makes meaningful connections between genotype and weather data that can be used by the breeder to inform breeding decisions and shorten breeding cycle length. The performer-based models can also be applied to other types of genomic selection such as salmon breeding for increased Omega-3 fatty acid production or similar animal husbandry applications. The code is available at: https://github.com/haakom/pay-attention-to-genomic-selection.

1. Introduction

Optimizing crop yield is a crucial step to enable secure and stable food production for the world. As the human population grows, the demand for food grows with it. To satisfy this demand, crops need to maximize production while keeping production costs and area needs to a minimum. Genomic Selection (GS) [6,27] has become a common approach to plant breeding in recent years. It is based on the use of genotypes through DNA variations, markers, coupled with known phenotype information from measured populations and uses this to predict the phenotype information in unknown populations. This approach can significantly reduce the length of breeding cycles [26]. However, this reduction depends entirely upon the models used for prediction. The phenotype of a crop is affected by a multitude of factors such as the crop genotype, the environment and the farmers management of the crop. This complexity is compounded by interactions between genotype and environmental factors. Environmental factors such as weather may also exhibit complex synergy effects that are difficult to model.

Traditional approaches to GS have used genomic best linear unbiased prediction (GBLUP) or Bayesian methods [3,28]. However, recent years have seen a shift towards the use of machine learning techniques for GS. With the recent success of artificial neural networks (ANNs) in other applications, such as image recognition and natural language processing (NLP) [21], many recent GS approaches employ similar methods [19,29]. ANNs are able to learn complex relationships between its inputs and outputs through simple weight update rules applied repeatedly with many input-output pairs. They treat the output, such as phenotype information, as an implicit function of its inputs, such as genotype and environmental information. This makes ANNs especially attractive for GS since they may be able to accurately model the complex dynamics and interactions between environmental factors and genotypes.

González-Camacho et al. [13] compared a multi layer perceptron (MLP) to a probabilistic neural network (PNN) on the task of classifying an individual's membership to a phenotypic class. They used 33 maize and wheat genomic and phenotypic datasets and found that the PNN outperformed the MLP measured using area under curve (AUC) and area under precision-recall curve (prAUC) on all datasets.

Khaki et al. [17] compare a convolutional neural network (CNN), combined with a recurrent neural network (RNN) to predict crop yields based on environmental data and management practices. Their model

[*] Corresponding author.
E-mail addresses: hakon.maloy@ntnu.no (H. Måløy), susanne.windju@graminor.no (S. Windju), stein.bergersen@graminor.no (S. Bergersen), muath.alsheikh@graminor.no (M. Alsheikh), keithd@ntnu.no (K.L. Downing).

https://doi.org/10.1016/j.atech.2021.100017
Received 24 June 2021; Received in revised form 9 October 2021; Accepted 12 October 2021

uses two CNNs to process weather and soil data respectively. The resulting output vectors of their CNNs were then fed into an RNN together with the historic average yield in sequence for prediction. They compare their method to a random forest (RF), MLP and LASSO. They use corn and soybean data from 13 states in the United States for the years 2016–2018. They find that their CNN-RNN framework substantially outperforms the other models achieving a root mean squared error (RMSE) of 9% and 8% of their respective average yields.

Barbosa et al. [2] proposed a CNN to learn the relevant spatial structures in five explanatory variables: nitrogen rate, seed rate, elevation map, soil's electroconductivity and satellite image. They then use these structures to model the yield response to nutrient and seed rate prescriptions for future predictions. They compare their model to a multiple linear regression (MLR) model, a fully connected network (FCN), an RF, and a support vector machine (SVM). Their model shows a reduction in test dataset RMSE of up to 68% when compared to multiple linear regression and up to 29% when compared to a random forest.

Although a plethora of studies applying ANNs to GS exist, most of these approaches use ANNs with large inherent inductive biases such as CNNs and RNNs. These networks make assumptions about the input such as local connectiveness for CNNs or sequential data in RNNs. However, genotype data generated through the use of markers do not necessarily fit these assumptions well. For instance, there is no inherent notion that marker positions that are physically close together in a sequence have more importance to each other than marker positions that are further apart. Similarly, although the genotype is organized as a sequence, there is not necessarily a sequential nature to the sequence. For instance, marker positions that follow each other are not inherently connected sequentially. Such inductive biases may therefore hamper model performance since they do not apply.

Recently, attention-based models have been applied to multiple applications with great success [9]. Common for these models is the use of self-attention mechanisms [10], which calculates the attention score of a position in a sequence, by attending to all positions in the same sequence. First introduced in [35], the transformer showed that a purely self-attentive model could produce state-of-the-art results on language translation tasks. This result sparked increased research on transformer-based models and has resulted in transformers producing the state-of-the-art performance in multiple domains [7,8,18,25]. However, transformers still suffer from an inability to process very long sequences due to the space complexity of the self-attention mechanism. Choromanski et al. [11] introduced the Performer, by substituting the softmax self-attention mechanism with a FAVOR+ mechanism. This enabled the Performer to handle very long sequences without the memory footprint of the traditional self-attention mechanism. However, this approach has not seen wide adoption and therefore the general limitations of the original transformer persist.

Transformer-based models make very few assumptions about the input data. Instead they use self-attention mechanisms to learn any underlying structure. Since these models introduce little inductive bias and instead learn the underlying dynamics and connectiveness of their inputs we hypothesize that they are better suited for GS than traditional ANNs. Due to the ability to visualize the self-attention maps within the model per input, they may also be more explainable than CNNs or RNNs, making it possible to use transformer-based models to inform the breeder of what dynamics affected the prediction. This could therefore further shorten the length of breeding cycles since the breeder could directly use the model to inspect potential marker interactions in the self-attention visualization.

In this work we are, to the best of our knowledge, the first to propose transformer-based models with the FAVOR+ mechanism, hereby referred to as performer-based models, for GS. We also firmly ground our proposed models in optimal configurations by doing a thorough search of each model's hyperparameter space using Bayesian optimization. We show that performer-based models significantly outperform other ANN architectures and traditional methods for GS. We also show that the performer-based models naturally handle multimodal input and leverage the different modalities for improved prediction performance. The performer-based models introduced in this work are general models that can be applied to other types of genomic selection such as salmon breeding for increased omega-3 fatty acids production or similar types of animal husbandry.

Our contributions are as follows:

- We introduce performer-based models to field of GS.
- We show that performer-based models outperform traditional ANNs for GS while needing significantly fewer trainable parameters.
- We propose a general performer-based model that natively handles multimodal input and is suitable for multi-domain GS.

2. Method

2.1. Dataset

The genomic data consisted of 2-row and 6-row barley genotypes. The genotype data was collected using single nucleotide polymorphisms (SNPs) measurements for 433 separate genotypes of barley. Each genotype contained 13321 SNP measurements. The phenotype data contained yield measurements from eight different locations in Norway for the years 2017 and 2018. We collected weather data from the same eight locations for both years in the time period 1st of May to 31st of August using The Norwegian Meteorological Institute FROST API[1] [15] and NIBIO Landbruksmeteorologisk Tjeneste (LMT) AgroMetBase.[2] The collected weather data contained the mean temperature for each day and the cumulative precipitation for each day.

2.2. Data processing

The data seen by the models contain two different data modalities, genotypes and weather data. The different modalities are therefore processed differently to prepare them for our models.

2.2.1. Genotype

Since the genotype data was originally encoded using SNPs, it consisted of sequences of nucleotides measured at different SNP positions, indicated by the symbols: "T", "C", "G", "A", "R", "S", "K", "Y", "M", "W", "failed", where "failed" indicates a failed reading. However, to make the genotypes readable by computers, we encode them as sequences of one-hot encoded vectors. This approach represents each nucleotide as one of the unit vectors in $\mathbb{R}^{11}$. We also add an "extra" symbol to make our unit vectors part of $\mathbb{R}^{12}$, since they are easier to work with in our models. Each genotype consists of a sequence of measurements at 13,321 SNP positions.

2.2.2. Phenotype

The phenotype data sometimes contained multiple yield measurements for the same genotype in one location. This was due to several plantings in the same experiment. Instead of creating separate genotype-yield pairs for each such case, we use the average yield for the genotype in each location. However, the distribution of yields for a genotype were sometimes heavily skewed, even in a single location. We therefore calculated the average using Eq. (1). This approach shifts the $yield_{avg}$ slightly towards the center of mass of the yield distribution and is therefore not as affected by outliers as the mean. The resulting data contained a single yield measurement per genotype per location. To make the yields fit within a sigmoid curve we also scaled the yields to a range between 0 and 1. After processing our yield data as described above, we were left

[1] https://frost.met.no/.
[2] https://lmt.nibio.no/.

Table 1
The Spearman rank correlation coefficient matrix for the collected weather data.

	Yield	Temperature	Precipitation
Yield	1.000	-0.381	0.452
Temperature	-0.381	1.000	-0.867
Precipitation	0.452	-0.867	1.000

with 2214 genotype-yield pairs from the entire period, 1275 of which were 2-row and 939 were 6-row variants.

$$yield_{avg} = \exp\left(\frac{1}{K}\sum_{i=0}^{K}\ln(yield_i)\right) \tag{1}$$

2.2.3. *Weather*

The crop locations were spread over large portions of Norway and therefore experienced a variety of climates. However, most crops were produced inland, with only one being produced in a coastal climate. To investigate whether a clear relationship between weather measurements and production yield exists, we calculated the Spearman rank correlation coefficient [33] in Table 1. From the table, no strong correlation between yield and either weather phenomenon can be observed. We created two versions of the weather data. The first was a mean weather dataset, containing the mean temperature and the mean precipitation for the entire measurement period for each location. The second was the raw historical data for each day. The two versions were processed using different methods.

1. *Mean weather* The mean weather contained the mean air temperature and mean precipitation for the entire period. The data was standardized separately for each modality using the standard scaler from Scikit-learn [30].
2. *Historical weather* Since the mean weather data only captured the mean of the weather over the entire period, it could lose days with abnormal temperature or precipitation measurements, since they may not significantly affect the mean. This was especially true if there were days with abnormally high and other days with abnormally low measurements. Such variations could cancel each other out in the mean calculation. However, they could have a significant impact on the yield of a crop. A historical weather record avoids this problem by capturing the daily variations of the weather and thus enables models to become sensitive to such changes and leverage them to better predict the yield. We therefore created a historical weather dataset which contained the mean air temperature and cumulative precipitation per day for the collected period. This gives a daily historical record of the weather data and is well suited for sequence analysis. However, since many days did not see any precipitation, the precipitation data was heavily positively skewed. To avoid the models becoming too sensitive to outliers we scaled the weather data using the robust scaler from Scikit-learn [30].

2.3. *Model architectures*

We compared two models types, using the same data and training regimes. The models were convolutional neural networks [20,22] and performers [11]. To develop a baseline with which to compare our models, we trained a model using Bayesian reproducing kernel Hilbert spaces regressions (RKHS) implemented in the BGLR package introduced by Perez and de los Campos [31].

2.3.1. *Convolutional neural networks*

The convolutional networks process the genotype sequence and weather data using separate networks. The genotype sequence is processed using one of two CNN architectures, while the weather data is processed using an simple MLP. This was done because the two data modalities do not easily mix. A CNN would therefore either have to learn specific filters in each layer of its pipeline that either only look at genotype data or only look at weather data. By separating the data processing into two pipelines we could instead train modality-specific networks that needed fewer model parameters to reach optimal performance. We then concatenated the output of the genotype pipeline to the output of the weather pipeline and fed the resulting vector into the prediction network. To explore the capabilities of CNNs we compared two possible architectures in the genotype pipeline.

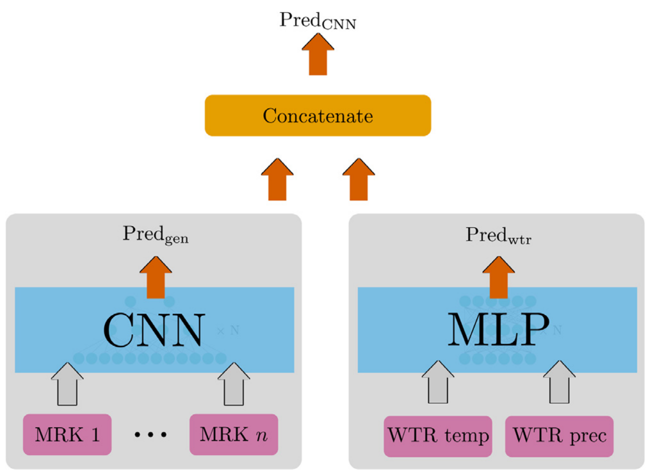


Fig. 1. The Vanilla CNN processes the genotype and historical weather data separately using a stack of residual blocks for the genotype data and an simple MLP for the weather data. The output vectors of the networks are then concatenated to form a single feature vector for the prediction network.

1. *Vanilla CNN* The first was a vanilla CNN, which consisted of a stack of one-dimensional convolutional layers with batch normalizations [16] between each layer. We used a kernel size of 6 and a stride of 2 for every layer. This reduces the output size of each layer, thereby increasing the receptive field of the following layer. The Vanilla CNN is visualized in Fig. 1.
2. *ResNet* The second CNN architecture was based on the ResNet architecture introduced in [14]. Rather than using a stack of regular convolutional layers, a ResNet architecture uses a stack of residual blocks. Each residual block consist some permutation of a two pathway information flow. The first pathway feeds the input through two convolutional layers. The second pathway, called the skip connection, feeds the input through a downsampling layer. This layer commonly uses a pooling layer or a convolutional layer with a kernel size 1 and a stride of 2. The output of the skip connection is added to the output of the second convolutional layer to form the output of a residual block. The skip connection introduces a shortcut for the gradient to take, resulting in the ability to train much deeper and expressive networks. However, the skip connection also explicitly discards information from the input. This may not affect performance much when dealing with data where spatially close inputs are statistically similar, such as images. However, for genotype information, such information loss could drastically reduce model performance. We therefore introduced a new residual block that explicitly avoided this loss of information. Our residual block fed its input through two convolutional layers with kernel size 6 and stride 2, similar to a regular residual block. Our skip connection fed the input directly to the output going through a downsample operation to make its dimensions correspond with the output dimension of the second convolutional layer. To preserve all the information from the input through the skip connection, the downsample operation padded two copies of the input with a zero-vector either at the end (Pad$^+$) or the beginning (Pad$^-$) of the sequence. It then used a convolutional layer

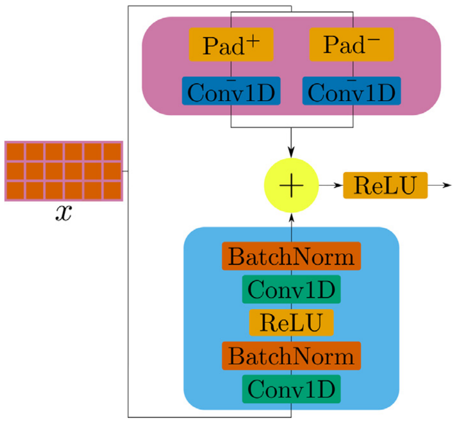


Fig. 2. The ResNet residual block. The input, x, follows two parallel processing pipelines. The downsample operation (top), downsamples the input through either right or left padding it, before it is fed through a 1D convolutional layer, using a kernel size of 1 and a stride of 2, with fixed weights at 1.0 and no bias ($\bar{\text{Conv1D}}$). The main processing pipeline (bottom) uses alternating 1D convolutional layers with a kernel size of 6 and a stride of 2, together with batch normalization layers to process the input. The pipelines are combined using a simple add operation before it is outputed by the residual block.

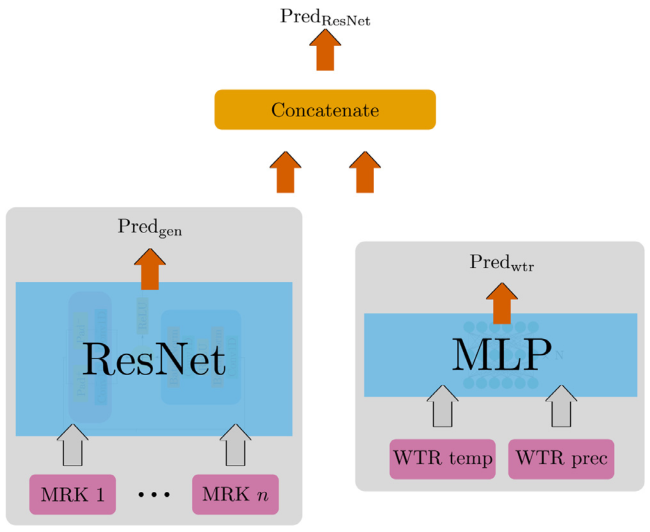


Fig. 3. The ResNet processes the genotype and historical weather data separately using a stack of residual blocks for the genotype data and a simple multilayer perceptron (MLP) for the weather data. The output vectors of the networks are then concatenated to form a single feature vector for the prediction network.

with a kernel size of 1 and stride 2 to downsample the input. The padding shifted the input so that the convolutional layer looked at two different subsets of the sequence. The convolutional layer had fixed weights of 1, no bias and no gradients. The two downsampled inputs were then added together to preserve the original information in a compressed representation. The compressed representation was then added to the output of the convolutional layers to complete the skip connection. This allowed a full representation original input to flow through every residual block, but with a reduced representation size, thereby increasing the receptive field of the following residual block. We visualize a residual block in Fig. 2 and the full ResNet architecture in Fig. 3.

2.3.2. *Performer networks*

Two of our performer-based models employed the same approach to the two data modalities as we described in Section 2.3.1. The genotype data and weather data were processed in separate pipelines before the output of the pipelines were concatenated to form a single vector that was then fed into the prediction network. A third model processed both genotype data and weather data in a single pipeline before the output was fed into the prediction network.

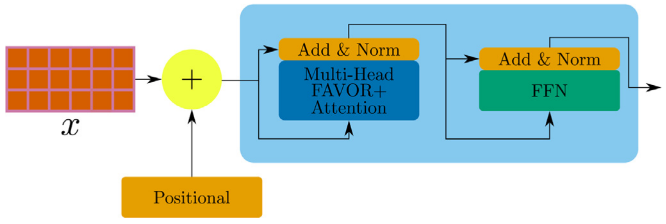


Fig. 4. A performer encoder module. The input (x), is positionally encoded before it is fed into a stack of encoder modules (blue). Here the positionally encoded input is passed through a Multi-Head FAVOR+ Attention mechanism sub-module and a Feed-Forward Network (FFN), sub-module with residual connections and normalization layers added between each sub-module.

Similar to [12], our performer-based models consisted of encoder modules stacked on top of each other, regularized with dropouts between each module [34]. Each encoder module contained a self-attention sub-module and a feed-forward network, where the self-attention sub-module traditionally calculates the attention score over its input using a softmax attention mechanism as described in [35].

However, the softmax attention mechanism has a space complexity of $O(L^2 + Ld)$, where L is the input sequence length and d is the dimension of the latent representation. The genotype sequences we used have $L = 13321$, which results in a space complexity that was not feasible given current hardware. We therefore substituted the softmax attention mechanism with the FAVOR+ mechanism described in [11], using the FastAttention implemented by [36], which approximates the softmax attention using kernel methods. This reduced the space complexity to $O(Lr + Ld + rd)$, where r is the number of random features sampled and $r << L$. Using the FAVOR+ mechanism we could construct performer architectures that were capable of processing very long sequences, thereby enabling their use in gene sequence processing.

As in [35], we configured the self-attention mechanism in a multi-head configuration to improve its performance. This configuration entails creating h parallel self-attention mechanism, dubbed heads, and linearly transforming the input for each head using a different weight matrix. This produced h different representations of the input, where h is the number of heads in the model. This configuration allowed each head to learn different attention scores. The multi-head attention scores were then concatenated and linearly transformed to compute the final attention score. To keep computational costs similar to that of a single-head approach, the dimensionality of each head was reduced to $\frac{input\_size}{h}$ using the linear transformation of the input.

Finally, we added a residual connection and a layer normalization to each sub-module in the encoder module. The residual connection added a shortcut through the network, providing an alternative route for the gradient to flow. This allowed deeper networks to be built since the error signal can always reach the lower levels in the network through the residual connections. We visualize a full performer encoder module in Fig. 4.

The FAVOR+ mechanism also allowed other kernels to be approximated. We therefore included the ability to approximate a ReLU kernel, dubbed “generalized attention” in [11], instead of the softmax attention in our performer models. We trained three different performer models:

1. *Plain performer* The first model used a performer network to process the genotype data, while it used an MLP to process the weather data. We used $r = 150$ in the performer FAVOR+ mechanism. This model is directly comparable to the convolutional networks presented in Section 2.3.1 and is visualized in Fig. 5. The concatenated output of

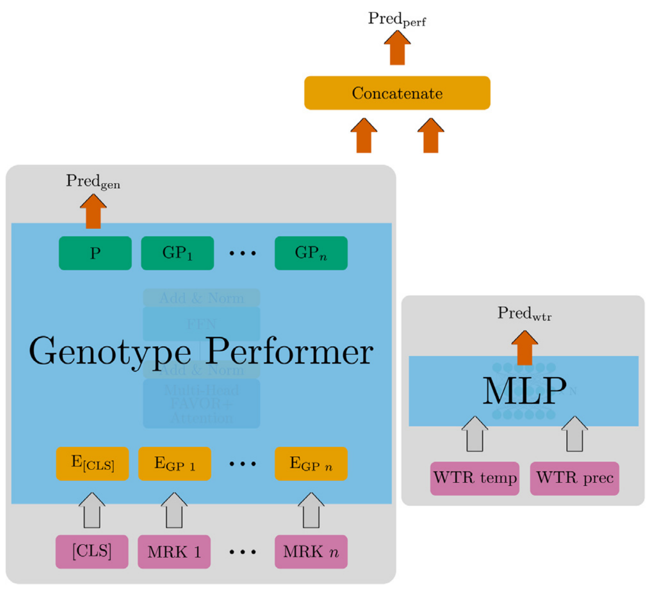


Fig. 5. The Plain Performer processes the genotype and historical weather data separately using a performer model for the genotype data, and a simple multilayer perceptron (MLP) for the weather data. The genotype data is positionally encoded before it is fed into the encoder stack. The output vectors of the networks are then concatenated to form a single feature vector for the prediction network. The [CLS] symbol is a special symbol added to the start of the input sequence and is the position used to generate the prediction vector P.

the performer network and the MLP were then fed into the prediction network.

2. *Historical performer* The second model used one performer network to process the genotype data and another performer network to process the weather data. We used $r = 150$ in the performer FAVOR+ mechanism in both models. This model also used the historical weather data, utilizing the sequential processing nature of performer networks. The Historical Performer is visualized in Fig. 6. The concatenated vector was then fed into the prediction network.
3. *Multimodal performer* The third model fully leveraged the lack of inductive bias in transformer models. For this network, we first upsampled the weather data to fit the dimensions of the genotype data and then concatenated the upsampled weather data to the end of the genotype sequence. This enabled the multimodal performer to attend to the genotype data and weather data simultaneously, allowing it to directly learn the relationship between genotypes and weather. We also used historical weather data for this model. We use $r = 300$ in the FAVOR+ mechanism to account for the increased sequence length. To further emphasize the two different data modalities to the performer, we added a constant of $+1$ to the weather vectors before the learned positional embedding was added. The full processing stack is visualized in Fig. 7. The output of the performer was then fed into a prediction network.

2.3.3. Prediction network

The prediction network consisted of either a single fully connected layer or an MLP, depending on whether it was processing the output from the Multimodal Performer or one of the other models respectively. The two layer version enabled the prediction network to interpret the two modalities before doing the prediction. Since the Multimodal Performer already handles the multimodal nature of the data, the prediction network only needed a single layer. The final layer of the prediction network contains a single predictive unit. The predictive unit computed a predicted scaled yield using a sigmoid activation function.

Table 2
The hyperparameters searched for the models, shown with the search space and distribution type.

Hyperparameter	Search Space	Distribution
Number of Conv Layers/Residual blocks[a]	1–10	uniform_int
Number of Filters per Conv layer[a]	16 –256	uniform_int
Number of Performer Layers[b]	1–4	uniform_int
Number of Performer Heads[b]	[4, 8, 16]	categorical
Number of Units in FFN[b]	[32, 64, 128]	categorical
Generalized Attention[b]	[True, False]	categorical
Number of Linear Layers[c]	1–6	uniform_int
Number of units per Linear layer[c]	16–32	uniform_int
Learning Rate	5e-2 - 1e-8	log scale
Dropout Rate	0.1–0.9	uniform
Number of Units in 1st Prediction Layer	1–128	uniform

[a] These hyperparameters are only used in convolutional models.
[b] These hyperparameters are only used in performer-based models.
[c] Linear Layers are not included in the Multimodal Performer.

2.3.4. BLGR model

The baseline model was trained using BGLR RKHS [31]. Since this model was unable to handle multi-channel input, we represented each nucleotide as a flotation point number, ranging from 0.0 to 1.0, where “failed” = 0.0, “T” = 0.1, “C” = 0.2,... and “W” = 1.0. We normalized the genotype through subtracting the mean and dividing by the standard deviation of the entire genotype dataset. The phenotype was represented as explained in Section 2.2.2.

2.4. Experimental details

Our experiment was separated into two parts. The first part aimed to find the best model architectures for each of our prediction models. The second part tested the best architectures we found in the first part, using a 10-fold cross-validation setup.

2.4.1. Finding optimal model architectures

To find the optimal model architectures we searched the hyperparameter space using the Optuna library [1]. This allowed us to search through large hyperparameter spaces efficiently by updating where a sample set of hyperparameters was drawn from using Bayesian priors. This approach naturally focuses the search process in areas with the most promising sets of hyperparameters rather than randomly sampling from the entire search space or using a naïve grid search approach. To set up our search we defined the hyperparameters to be included in the search and defined their ranges. The hyperparameters and their search ranges are shown in Table 2. In addition we searched through a set of hyperparameters that were common for all models. We ran 400 trials per model search, sampling hyperparameters using a multivariate TPE sampler [4,5]. Each trial was allowed to run for up to 128 epochs, but to avoid spending too much time exploring unpromising trials, we used a hyperband pruner [23] with min_resources = 3, max_resources = ‘auto’ and a reduction_factor of 3. All models were trained using the AdamW optimizer [24]. The search was conducted using the NTNU IDUN computing cluster [32]. The trials were run in parallel on 10 nodes, each consisting of two Intel Xeon cores and either 2 NVIDIA Volta V100 GPUs or 2 NVIDIA Pascal P100 GPUs. A complete search for a single model required from ≈ 24 hours to several days to finish.

2.4.2. 10-fold cross-validation

To test the model architectures resulting from the hyperparameter searches, we employed a 10-fold cross-validation approach. This entailed dividing our dataset into 10 equally sized partitions and then using 9 partitions to train a model and the 10th partition to test the resulting model. This was repeated using each partition as a test partition only once. Each test fold was randomly sampled to ensure it being a representative sample. This approach enabled us to report test

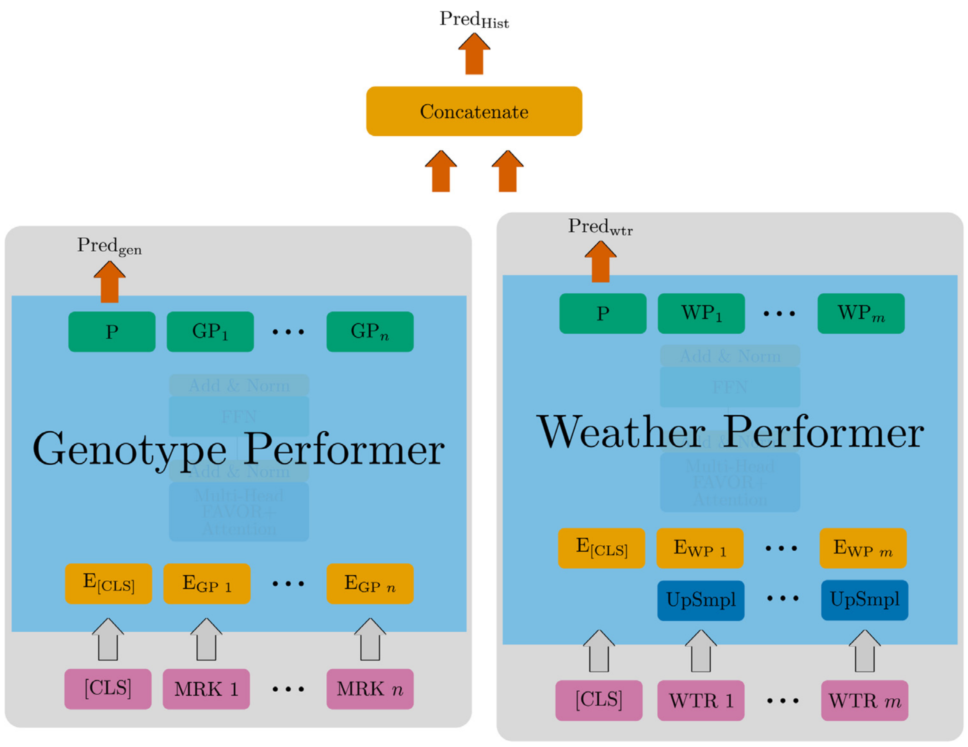


Fig. 6. The Historical Performer processes the genotype and historical weather data separately using one performer model for each modality. The genotype data is positionally encoded before it is fed into the encoder stack, while the weather data is also upsampled before the positional encoding is added. The output vectors of the two performers are then concatenated to form a single feature vector for the prediction network. The [CLS] symbol is a special symbol added to the start of the input sequence and is the position used to generate the prediction vector P.

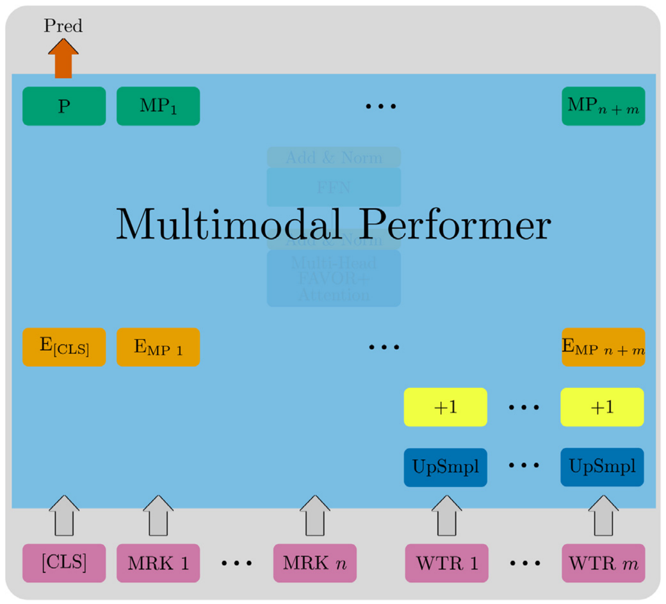


Fig. 7. The Multimodal Performer takes in both the genotype sequence and the weather sequence. It then upsamples the weather sequence to fit the dimensions of the genotype sequence. It then further enhances the difference in modalities by adding a 1 to every measurement in the weather sequence. It then concatenates the two sequences and positionally encodes the entire sequence before it feeds the sequence through the encoder stack. The [CLS] symbol is a special symbol added to the start of the input sequence and is the position used to generate the prediction vector P.

Table 3
The 10-fold cross-validation results for each of the models. We report the R^2 score, the Mean Average Error (MAE) and the Root Mean Squared Error (RMSE) for each model.

Model	R^2score	MAE	RMSE
BGLR RKHS	0.076	128.77	149.78
Vanilla CNN	0.807	55.05	71.63
ResNet	0.690	72.55	90.79
Performer	0.815	54.61	70.34
Historical Performers	**0.820**	53.61	69.29
Multimodal Performer	**0.820**	**53.11**	**69.05**

results on the entire dataset, giving a better evaluation of each model's performance than if we used a single randomly sampled test set. For each test partition the model architecture found in the hyperparameter search was initialized with the same set of starting parameters using a seeded random sampler. This starts the model off with randomly initialized variables, but ensures the same set of parameters for each fold. This resulted in the only changing variable being the 9 partitions used for training. During training, we used 20% of the training data as a validation dataset. Each model was trained for 128 epochs. For testing, we first loaded the set of parameters that achieved the lowest validation loss before we ran the test set through the model to find its test performance. For the BLGR model, we employed the same approach, however, each model was trained using 18K iterations with a 3K burn-in.

3. Results

In the cross-validation test process we tested each architecture, using the optimal set of hyperparameters found through the hyperparameter search. An overview of the best hyperparameter combinations for each model is shown in Appendix A. We report the R^2 score, Mean Average Error (MAE) and the Root Mean Squared Error (RMSE) for each model. The results are shown in Table 3. We found that the Performer models were superior to both the convolutional models and the BGLR RKHS model. This is also visible in Fig. 8, where we visualize the RMSE for each cross-validation fold per model. Additionally, among the performer-based models, we found that the Multimodal Performer slightly outperformed the two other models. This is also visible in the

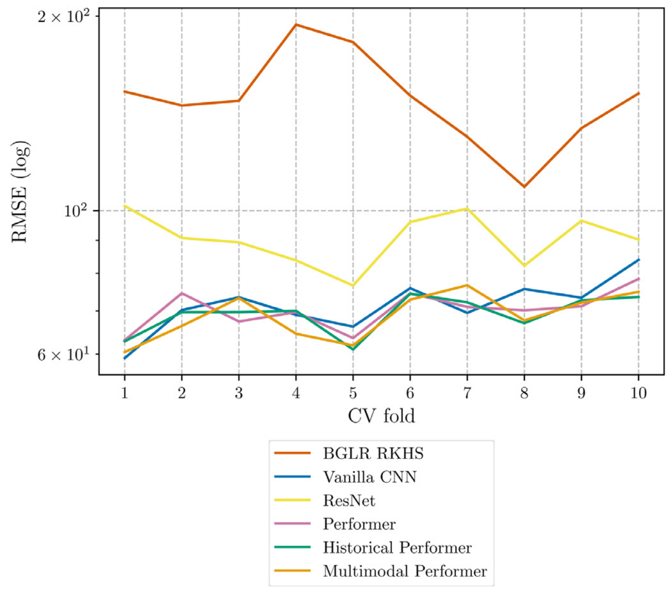


Fig. 8. We show the root mean squared error (RMSE) for each of the 10 cross-validation (CV) folds for each model. The performer-based models consistently outperform the other models except for folds 1 and 7, where the Vanilla CNN only slightly outperform the performer-based models.

residual box plot shown in Fig. 9, where the Multimodal Performer shows the smallest spread of all models.

3.1. Model analysis

3.1.1. Model fit

To asses the fit of our models we analyze the testing residual plots for the BGLR RKHS, the Vanilla CNN and the Multimodal Performer. The plots are shown in Fig. 10. The BGLR RKHS model shows some pattern in the residuals, indicated by the dip in the red line, while both the Vanilla CNN and Multimodal Performer shows little pattern in the residuals. We also observe significantly lower and constant variances in residuals for the Vanilla CNN and Multimodal Performer than we do in the BGLR RKHS model, explaining the large difference in R^2 scores. However, all models also show the possible presence of outliers, indicated by individual points with particularly high variance around $\hat{y} \approx 400$ and $\hat{y} \approx 800$ for the Vanilla CNN and Multimodal Performer and around $\hat{y} \approx 500$ and $\hat{y} \approx 600$ for BGLR RKHS. Again, these points are much more obvious in the BGLR RKHS model, explaining the reduced performance. The lack of patterns and presence of constant variance in the residuals of the Vanilla CNN and Multimodal Performer indicate a good fit for both models.

3.1.2. Trainable parameters

In Fig. 11 we show the number of trainable parameters for each of the five non-Bayesian models. The three performer-based models had the lowest number of trainable parameters. The Vanilla CNN had three times more trainable parameters than the performer-based models and the ResNet had an order of magnitude more trainable parameters than the Vanilla CNN. It is therefore possible to argue that these models had a high capability of overfitting to the training data, which could explain their lower performance on the test set. It is therefore encouraging that all performer-based models are significantly smaller than the CNNs. These smaller model sizes are suggestive of the generalization capabilities of performers, since they need fewer parameters to achieve similar or better performance. It also suggests that the reduction in inductive biases is beneficial, since it allows the models to learn the underlying correlations in the data instead of relying on assumptions coded into the model structure.

3.1.3. Seasonal data

In Section 2.2.3, we describe that our dataset contained weather data from 2017 and 2018. However, historical weather records show that 2018 was a particularly hot year, with abnormally low precipitation rates. This could mean that the Multimodal Performer was poorly equipped to handle data from other time periods, since it may have learned correlations that are particular to 2018. To investigate this we quantified the performance difference in Table 4. From the table, it is clear that the model performs better on 2017 data than on 2018 data, indicated by superior performance across all metrics. This suggests that the model has learned the true underlying factors contributing to pro-

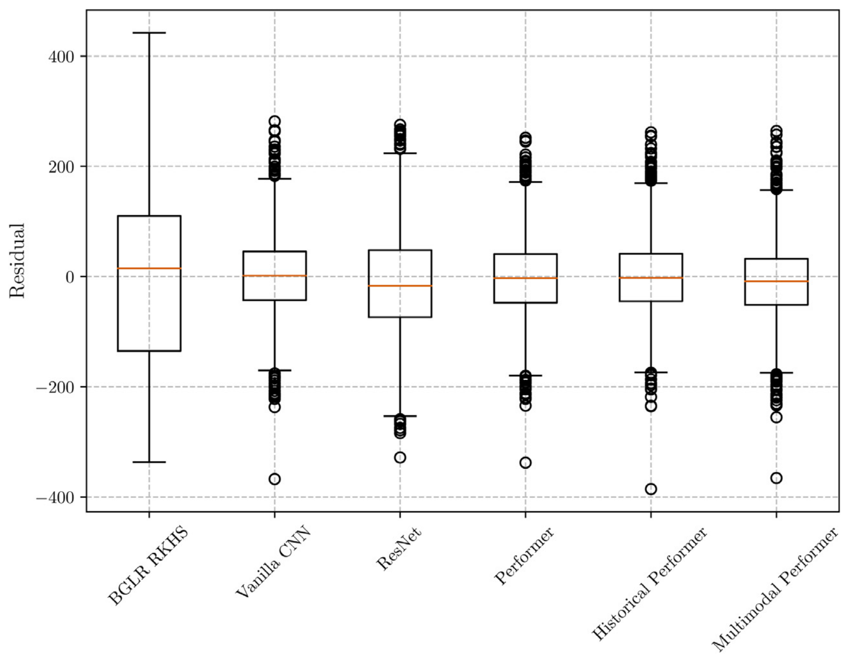


Fig. 9. The box plot shows the residual performance of our models visualizing the 0th percentile (lower whisker), the first quartile (lower part of box), the median (center line), the third quartile (upper part of box) and the 100th percentile (top whisker). Outliers are visualized as circles. The highest spread is observed for the BGLR RKHS model, while the lowest spread is observed for the Multimodal Performer.

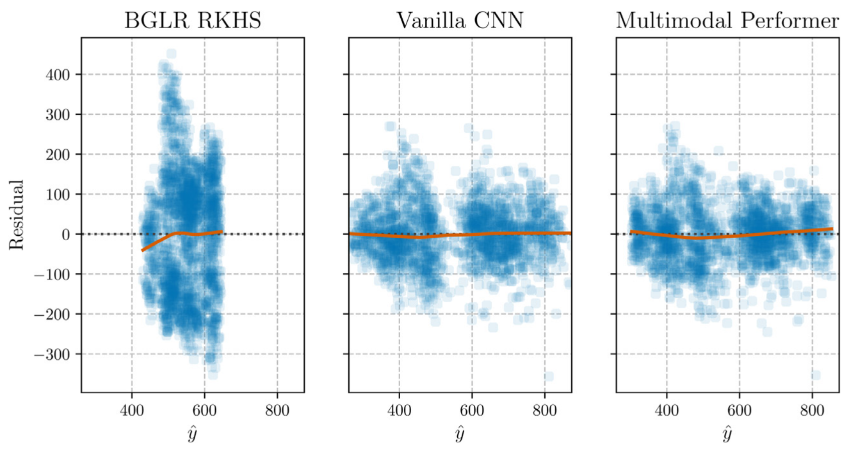


Fig. 10. The residual plots of three models: BGLR RKHS, Vanilla CNN and Multimodal Performer. The red line is a smooth fit to the residual, intended to make it easier to identify a trend. There is some pattern present in the BGLR RKHS model as seen by the dip in the smoothed fit. For the Vanlilla CNN and Multimodal Performer there is little pattern in the residuals. It is also clear that the BGLR RKHS has a much larger variance than the Vanilla CNN and Multimodal Performer.

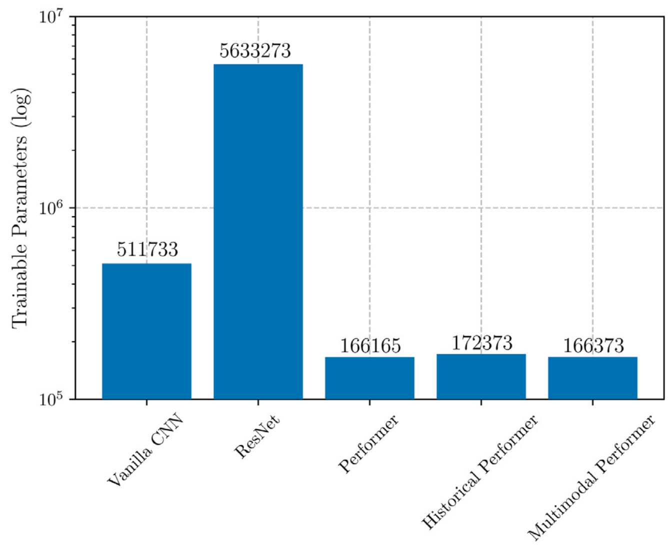


Fig. 11. The five non-Bayesian models differ in the number of trainable parameters in the optimal model found. The ResNet has an order of magnitude more trainable parameters than the Vanilla CNN, while the Performer-based models have three times fewer parameters than the Vanilla CNN.

Table 4
The Multimodal Performer test data performance for the years 2017 and 2018. We report the R^2 score, the Mean Average Error (MAE) and the Root Mean Squared Error (RMSE).

Year	R^2 **score**	MAE	RMSE
2017	**0.683**	**45.26**	**58.92**
2018	0.627	58.65	75.68

Table 5
The Multimodal Performer test data performance for the 2-row and 6-row barley variants. We report the R^2 score, the Mean Average Error (MAE) and the Root Mean Squared Error (RMSE).

Variant	R^2 **score**	MAE	RMSE
2-row	0.759	55.10	71.85
6-row	**0.856**	**50.71**	**65.93**

duction yields, whereas abnormal weather data confuses the model and leads to deteriorated performance.

3.1.4. 2-row and 6-row comparison

In Section 2.1 we stated that our dataset contained both 2-row and 6-row barley genotypes. Since model performance may be different for the two barley variants we compared 2-row and 6-row variants results on the test data using the Multimodal Performer. In Table 5 we show the Multimodal Performer performance on the two barley variants. It is clear that the model performs much better on 6-row than it does on 2-row. This is particularly interesting since our dataset contained fewer 6-row examples than 2-row examples. This should intuitively lead to better performance on 2-row since the model would have more examples to learn from. We therefore show the true yield densities for the two barley variants along with the predicted yield densities in Fig. 12. In the figure it is clear that the 2-row variant has two pronounced modes, while the 6-row has a slight tendency towards three modes. The Multimodal Performer is able to capture the modes of both variants, but one of the predicted 2-row modes is shifted to the right and has a much higher density than the true distribution, explaining the reduction in performance.

3.1.5. Self-attention visualizations

In this section we illustrate the self-attention maps produced by the Multimodal Performer model. Visualizing these maps can help to better understand how the model interprets the data and what types of input have the most impact on the output of the model. We start by visualizing the self-attention maps for the genotype portion of the input sequence. In Fig. 13 we visualize 5 attention heads of the genome producing the highest measured yield in the test set of the model (Genome 1) as well as the genome producing the lowest measured yield form the same location (Genome 2). We have also verified that the model does predict high and low yield values respectively for the different genotypes. From Fig. 13, it is clear that the model can identify particular SNP positions as more important than others. This is indicated by global attention activations (vertical lines) in the different heads. Such attention activations suggests that the model weights the nucleotides at these SNP positions as particularly important for the yield prediction. This could indicate that the Multimodal Performer is able to model interactions between SNP positions, making it especially useful for GS. Moreover, each head seems to have learned to pay attention to different SNP positions, suggesting that the model utilizes its parallel processing capabilities and has learned multiple representations of the data.

We now visualize a small square of the weather portion of the input sequence. We visualize every day of weather measurements along the x-axis, but only the first 123 SNP positions along the y-axis to make the figures easier to read. In Fig. 14 we show the attention activations for

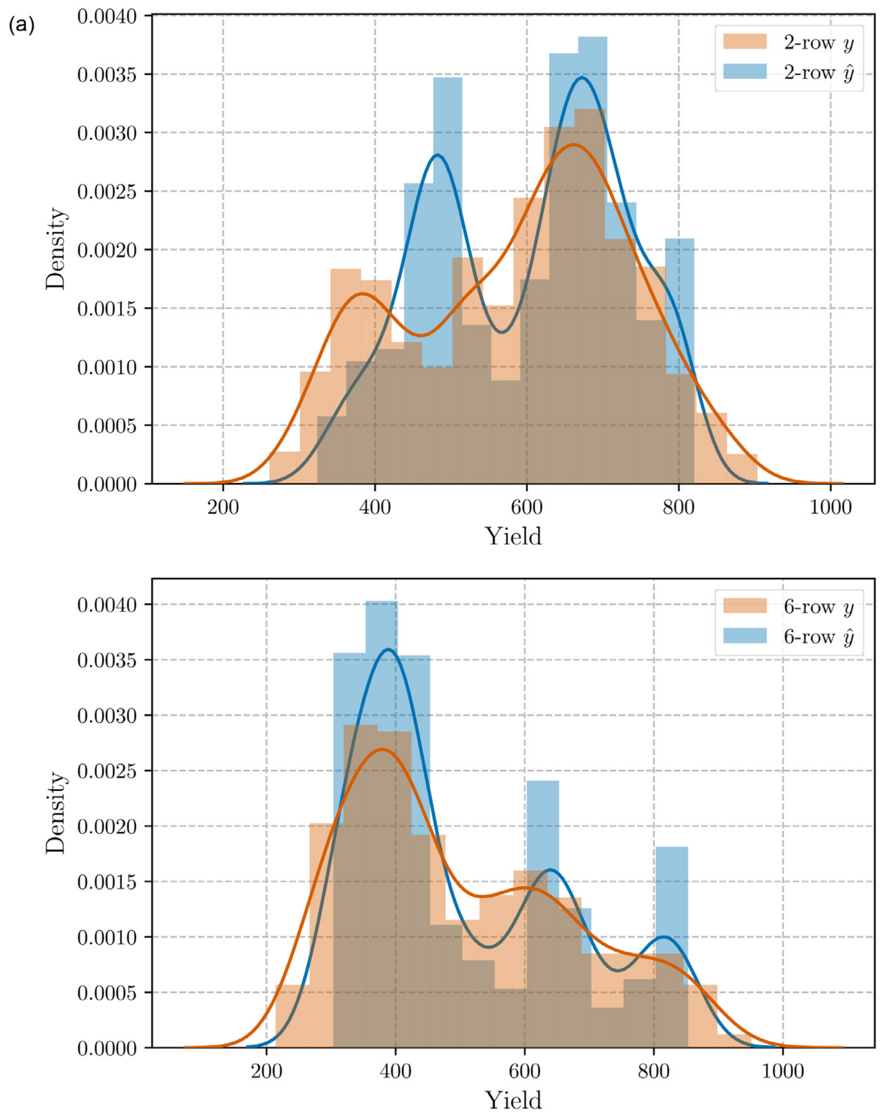


Fig. 12. The yield density distributions for 2-row and 6-row barley variants together with the Multimodal Performer predictions for each variant. The model is able to capture the multimodal distibutions of both variants, but one of the modes in the 2-row variant is shifted to the right.

three of the heads in the Multimodal Performer and overlay the weather data as a bar plot along the x-axis. Here, the parallel processing capabilities are even more apparent, since the different heads have clearly learned to pay attention to different weather phenomenon. Head 1 pays attention to temperature, indicated by strong global attention activation (vertical lines) aligning with higher measurements for temperature. Heads 4 and 5, on the other had, seem to pay attention to precipitation, again indicated by global attention activations aligning with high precipitation measurements. This is especially clear in the very strong activations aligning with the two days with abnormally high precipitation measurements, indicating that these two days are particularly important for the model yield prediction.

We also visualize the self-attention map for the weather portion using the same two genomes as in Fig. 13. In Fig. 15 we visualize the genome producing the highest yield (left) and the genome producing the lowest yield (right) on the same location. In the figure, the general self-attention maps look similar, however, for the low-yield genome we see strong activations along many of the SNP positions (y-axis) than in the high-yield genome. This indicates that the model learns to weight weather data differently, based on the particular genome fed in with the weather data, suggesting that the model is able to model how weather phenomena affect yield for different genotypes.

4. Discussion

We have introduced performer-based models to GS and shown that they are able to learn meaningful representation while outperforming more traditional approaches applying ANNs to GS. Through an extensive search of hyperparameters using Bayesian optimization techniques we found an optimal set of hyperparameters for our models. We then compared each optimally hyperparametrized model on crop yield prediction using barley genotype and weather data sourced from 8 locations over two years of production. Our results suggests that performer-based models outperform other ANN approaches by a significant margin.

The use of performer-based models has several advantages over traditional ANN approaches for GS. First, performers have less inductive bias than traditional ANNs. This enables performers to learn underlying correlations more accurately and naturally than traditional ANNs, which to a larger degree rely on such correlations being encoded into the model architecture. Second, performer-based models are multimodal by nature. CNNs and RNNs were developed with particular applications in mind, they are therefore less effective for applications that are significantly different. The transformer was instead introduced as a generalization of the MLP. It is therefore able to handle all modalities of data with relative ease, which has also been shown in the literature. Unlike the CNN models, which must be coupled with a MLP to handle the two

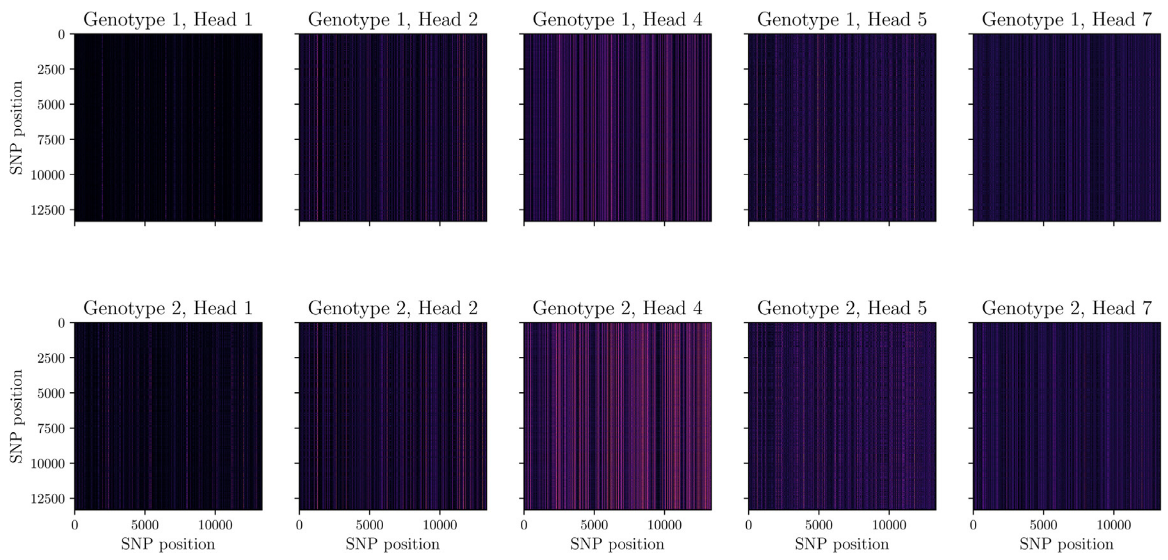


Fig. 13. The self-attention map of the genotype portion of the input shows how each SNP position pays attention to every other SNP position in the Multimodal Performer. It is visualized by plotting each SNP position against all SNP position on both the x- and y-axis. We show the 5 most visually salient heads for the same model being fed the genome producing the highest measured yield in the test set (top row), and the genome producing the lowest measured yield on the same location (bottom row). The model finds particular SNP positions more important than other, indicated by global attention activations (vertical lines) corresponding to their position on the x-axis. This indicates that the values at these SNP positions contribute more than other SNP position values when the model predicts the yield.

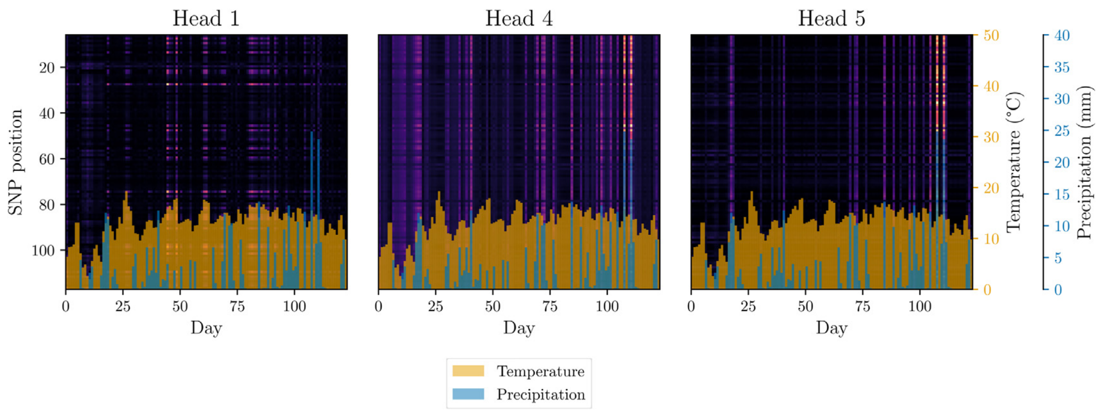


Fig. 14. Multimodal Performer self-attention maps for the weather portion of the input sequence. Weather measurements are overlayed on the x-axis to show correlations between weather data and attention activations. The y-axis shows the SNP position and the x-axis shows the day number for the weather data. The global attention activations (vertical lines) align well with precipitation measurements in Heads 4 and 5, indicating that these heads have learned to pay particular attention to such measurements during prediction. In Head 1, global attention activations align more with high temperature measurements albeit less prominent in the first month.

modalities of our data, the Multimodal Performer handles both modalities in a single network architecture. It also requires little extra processing of the two modalities other than making sure they are of the same dimensionality, making the Multimodal Performer the easier model to implement and use. Third, the self-attention maps of performer-based models can be visualized during prediction to show what the model is “paying attention to”. As we showed in Section 3.1.5, such visualizations can be highly informative and intuitive to interpret as opposed to many visualization techniques commonly applied to ANNs. The observation that different genotypes produce different self-attention activations for the same weather data aslo suggests high potential for its usage in GS. It could, for example, be possible to use the Multimodal Performer to find the optimal genotype for a particular location, using average weather data over many years in combination with yield predictions from the model. The breeder could also directly use how self-attention maps visualize SNP position interactions to inform future selection and breeding of genotypes. Such approaches could drastically shorten the breeding cycle length. The generality of the performer-based models also enable their use beyond crop yield prediction. They could, for instance, be used to inform selection of salmon genes in salmon farming for improved production of Omega-3 fatty acids or in other animal husbandry applications.

Although we show many positive results, our approach also has some drawbacks. The main drawback is the size of our dataset. In Section 2.2.2 we mention that our dataset only contained 2214 genotype-yield pairs. Traditionally, this is not considered a big dataset. However, since the Multimodal Performer is also a comparatively small model it suggests that the performance we report on our dataset is indicative of the model’s performance on new data and that it is not merely a result of the model overfitting. We would still like to stress that more data is likely to further improve model performance, especially considering that one of the two years available in our data was an abnormal year in terms of environmental factors. We could also have tested the model on several of the available datasets to better gauge the performance of the model, but these datasets neither contain the same SNP markers as our own, nor the same types of environmental data. They would therefore require models to be adapted to work with the new data and thereby result in a new model for each new dataset. Such test-

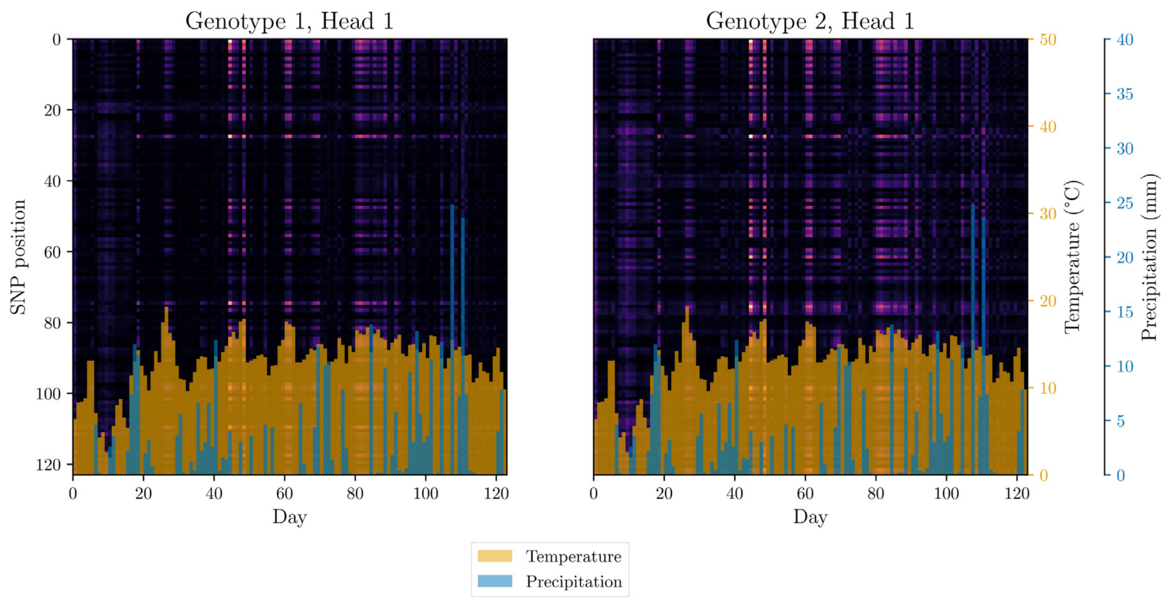


Fig. 15. Multimodal Performer self-attention maps for the weather portion of the input sequence using two different genomes. The y-axis shows the SNP position and the x-axis shows the day number for the weather data. The attention activations for Genotype 2 (right) are stronger for many SNP positions than those for Genotype 1 (left), indicating that the model pays attention to different things, given the same location, but across genotypes.

ing is still very interesting and could be used to further evaluate the Multimodal Performer on multiple datasets in the future.

Another aspect of our dataset that could have resulted in improved performance is the environmental factors used. In Section 2.2.3 we state that we used both mean temperature and cumulative precipitation for each day as our weather data. However, there are additional environmental factors that could have been used: soil data, solar radiation and vapor pressure. Such environmental data could give the Multimodal Performer further information on what factors contribute to yield and therefore improve model performance. Since our weather data was collected through the FROST API and NIBIO LMT AgroMetBase, and not the farms themselves, such information was not available. It would therefore be interesting to further study how extra environmental information affects model performance.

A third importand aspect of our dataset was that genoytpes and weather data were present in both training and test data. In Section 2.2 we describe how the dataset was processed and that the final dataset contained 2214 genotype-yield pairs. The process we describe resulted in the same genotype being tested in multiple locations. When we divided our dataset into 10 folds for the cross-validation runs, this also means that both genotypes and weather data were present in both the training data and the test data. However, any combination of genotype and weather data is only present in either the training data or the test data. This still means that the models have seen both the genotype and weather data during training, however independently. Although this might seem like a breach of the standard approach to splitting data into training and validation sets, we argue that this is not the case. First, the same splits were seen by every model, making the model comparisons valid. Second, this setting is good proxy for what might happen in real world GS. Next year's trials may contain many of the genotypes included in a previous year, thereby producing a similar situations to what we have in our dataset splits. However, weather data will likely change. It would therefore be valuable to test the model using previously unseen weather data.

We mention in Section 2.2.2 that the phenotype values of our dataset were scaled to fit between 0 and 1 to facilitate the use of sigmoid activation functions in our models. This choice means that the extreme phenotype values of our dataset will be pushed to the far left and far right tails of the sigmoid function. This has the unfortunate effect of restricting the variance of the models prediction distributions since it is not possible for the model to predict a value that is larger than the highest phenotype measurement. This effect can also be seen in Fig. 12. To avoid this issue, the scaling of phenotype values should be scaled using a range that is slightly larger than the minimum and maximum phenotype values in the future. This will shift the resulting distribution towards the middle of the sigmoid curve, thereby allowing the model to over- and underestimate its prediction during training and learn to fit the entire distribution.

One potential limiting factor in the design of our performer-based models was the size of the output head. In Section 2.3 we stated that the dimensionality of the model input was 12. Due to how the performer was built, this also means that the final prediction head produced a prediction vector of the same dimensionality. In the performer-based models this vector is responsible for representing the entire processing of the model. Given that the model processeed an input sequence of length 13445, each element of which was 12-dimensional, a single 12-D vector was a very compressed representation of the inner workings of the model. This could also help explain why the optimal architecture found for the Multimodal Performer contained only one layer, since multiple layers could create inner representations too complex to be expressed by a 12-dimensional vector. It would therefore be very interesting to increase the expressiveness of the model by increasing the size of the prediction vector. This could be done by simply upsampling the entire sequence to a higher dimension in the input layer, similar to what we do for the historical weather sequence in the Multimodal Performer. Such studies could further increase model performance and uncover even more complex representations by analyzing the resulting self-attention maps.

5. Conclusion

In this paper we introduced performer-based architectures to the field of genomic selection with the application of crop yield prediction. We showed that these architectures outperformed convolutional neural network architectures as well as Bayesian approaches on the task of yield prediction from SNP-based genotypes and weather data. We also showed that the multimodal capabilities of performer-based architectures made the models easier to implement and could further increase performance by processing multiple modalities simultaneously, using a self-attention mechanism. Through analyzing the self-attention maps produced in the performer-based models we showed that the models learned to pay attention to different weather phenomenon in different attention heads and correlated SNP positions with each other to produce its predictions. We also showed that the lack of inductive bias in performer-based models improved model generality and drastically reduced the number of trainable parameters needed to achieve high performance.

We suggest that future work should explore applying performer-based models to several readily available genomic selection datasets. We also suggest that these models should be applied to other applications within genomic selection, such as genotype selection for salmon farming or other types of animal husbandry. Such studies could reveal the full potential of performer-based models and help reduce breeding cycles.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Susanne Windju reports a relationship with Graminor AS that includes: employment.

Stein Bergersen reports a relationship with Graminor AS that includes: employment.

Muath Alsheikh reports a relationship with Graminor AS that includes: employment.

CRediT authorship contribution statement

Håkon Måløy: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Susanne Windju:** Resources, Data curation. **Stein Bergersen:** Resources, Data curation. **Muath Alsheikh:** Resources, Data curation. **Keith L. Downing:** Writing – review & editing, Supervision.

Acknowledgments

We would like to thank Graminor for providing the genotype-yield dataset for this research. We would also like to thank Dr. Ekrem Misimi for his helpful feedback during the writing of this paper. This work was supported by the Center for Research-based innovation in Aquaculture Technology, Exposed Aquaculture Operations (RCN 237790).

Appendix A. Hyperparameter search results

Table A1
The best hyperparameter combination for the Vanilla CNN. In brackets, the index corresponds to the layer number.

Hyperparameter	Value
Dropout Rate	0.22
Learning Rate	0.008
Weight Decay	0.0045
Number of Convolutional Layers	10
Number of Filters per Layer	[256, 117, 79, 143, 101, 42, 16, 67, 117, 17]
Number of Linear Layers	3
Number of Units per Layer	[21, 17, 17]
Number of Units in Prediction Layer	28

Table A2
The best hyperparameter combination for the ResNet. In brackets, the index corresponds to the layer number.

Hyperparameter	Value
Dropout Rate	0.15
Learning Rate	0.001
Weight Decay	0.010
Number of Convolutional Layers	14
Number of Filters per Layer	[100, 234, 39, 255, 137, 254, 222, 23, 250, 238, 35, 243, 195, 206]
Number of Linear Layers	5
Number of Units per Layer	[28, 29, 30, 26, 29]
Number of Units in Prediction Layer	118

Table A3
The best hyperparameter combination for the performer.

Hyperparameter	Value
Dropout Rate	0.30
Learning Rate	0.0024
Weight Decay	0.02
Number of Performer Layers	1
Number of Performer Heads	16
Head Dimension	2
Number of Units in FFN	128
Generalized Attention	True
Number of Linear Layers	1
Number of Units per Linear Layers	[28]
Number of Units in Prediction Layer	29

Table A4
The best hyperparameter combination for the Historical Performer. In brackets, the leftmost value is the performer processing the genome and the rightmost value is the performer processing the weather data.

Hyperparameter	Value
Dropout Rate	0.16
Learning Rate	0.0003
Weight Decay	0.002
Number of Performer Layers	[1, 1]
Number of Performer Heads	[16, 2]
Head Dimension	[2, 8]
Number of Units in FFN	[128, 128]
Generalized Attention	[True, True]
Number of Units in Prediction Layer	69

Table A5
The best hyperparameter combination for the multimodal performer.

Hyperparameter	Value
Dropout Rate	0.20
Learning Rate	0.0007
Weight Decay	0.003
Number of Performer Layers	1
Number of Performer Heads	8
Head Dimension	8
Number of Units in FFN	64
Generalized Attention	True

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[2] A. Barbosa, R. Trevisan, N. Hovakimyan, N.F. Martin, Modeling yield response to crop management using convolutional neural networks, Computers and Electronics in Agriculture 170 (2020) 105197.

[3] P. Bellot, G. de Los Campos, M. Pérez-Enciso, Can deep learning improve genomic prediction of complex human traits? Genetics 210 (3) (2018) 809–819.
[4] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, 25th annual conference on neural information processing systems (NIPS 2011), volume 24, Neural Information Processing Systems Foundation, 2011.
[5] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: International conference on machine learning, PMLR, 2013, pp. 115–123.
[6] R. Bernardo, Prediction of maize single-cross performance using rflps and information from related hybrids, Crop Science 34 (1) (1994) 20–25.
[7] G. Bertasius, H. Wang, L. Torresani, Is space-time attention all you need for video understanding? Proceedings of the 38th International Conference on Machine Learning 139 (2021) 813–824.
[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
[9] S. Chaudhari, V. Mithal, G. Polatkan, R. Ramanath, An attentive survey of attention models, arXiv preprint arXiv:1904.02874 (2019).
[10] J. Cheng, L. Dong, M. Lapata, Long short-term memory-networks for machine reading, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 551–561, doi:10.18653/v1/D16-1053.
[11] K.M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J.Q. Davis, A. Mohiuddin, L. Kaiser, D.B. Belanger, L.J. Colwell, A. Weller, Rethinking attention with performers, in: International Conference on Learning Representations, 2021. https://openreview.net/forum?id=Ua6zuk0WRH
[12] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186, doi:10.18653/v1/N19-1423.
[13] J.M. González-Camacho, J. Crossa, P. Pérez-Rodríguez, L. Ornella, D. Gianola, Genome-enabled prediction using probabilistic neural network classifiers, BMC genomics 17 (1) (2016) 1–16.
[14] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016), doi:10.1109/cvpr.2016.90.
[15] M. Institutt, Frost, 2020. https://frost.met.no.
[16] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.
[17] S. Khaki, L. Wang, S.V. Archontoulis, A cnn-rnn framework for crop yield prediction, Frontiers in Plant Science 10 (2020) 1750.
[18] S. Khan, M. Naseer, M. Hayat, S.W. Zamir, F.S. Khan, M. Shah, Transformers in vision: a survey, CoRR (2021).
[19] T. van Klompenburg, A. Kassahun, C. Catal, Crop yield prediction using machine learning: a systematic literature review, Computers and Electronics in Agriculture 177 (2020) 105709.
[20] Y. LeCun, Generalization and network design strategies, Connectionism in perspective 19 (1989) 143–155.
[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444, doi:10.1038/nature14539.
[22] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.
[23] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: a novel bandit-based approach to hyperparameter optimization, The Journal of Machine Learning Research 18 (1) (2017) 6765–6816.
[24] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, ICLR, 2019.
[25] H. Måløy, Echobert: a transformer-based approach for behavior detection in echograms, IEEE Access 8 (2020) 218372–218385, doi:10.1109/ACCESS.2020.3042337.
[26] T. Meuwissen, B. Hayes, M. Goddard, Accelerating improvement of livestock with genomic selection, Annu. Rev. Anim. Biosci. 1 (1) (2013) 221–237.
[27] T.H. Meuwissen, B.J. Hayes, M.E. Goddard, Prediction of total genetic value using genome-wide dense marker maps, Genetics 157 (4) (2001) 1819–1829.
[28] O.A. Montesinos-López, A. Montesinos-López, J. Crossa, F.H. Toledo, J.C. Montesinos-López, P. Singh, P. Juliana, J. Salinas-Ruiz, A bayesian poisson-lognormal model for count data for multiple-trait multiple-environment genomic-enabled prediction, G3: Genes, Genomes, Genetics 7 (5) (2017) 1595–1606.
[29] O.A. Montesinos-López, A. Montesinos-López, P. Pérez-Rodríguez, J.A. Barrón-López, J.W. Martini, S.B. Fajardo-Flores, L.S. Gaytan-Lugo, P.C. Santana-Mancilla, J. Crossa, A review of deep learning applications for genomic selection, BMC genomics 22 (1) (2021) 1–23.
[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.
[31] P. Perez, G. de los Campos, Genome-wide regression and prediction with the bglr statistical package, Genetics 198 (2) (2014) 483–495, doi:10.1534/genetics.114.164442.
[32] M. Själander, M. Jahre, G. Tufte, N. Reissmann, EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019. 1912.05848
[33] C. Spearman, The Proof and Measurement of Association Between Two things, International Journal of Epidemiology 39 (5) (2010) 1137–1150, doi:10.1093/ije/dyq191.
[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5998–6008. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
[36] P. Wang, performer-pytorch (2020). (https://github.com/lucidrains/performer-pytorch)