Trait Prediction using Deep Learning

Mowlaei, M. Erfan * Biswas, Sanchari †

April, 2022

Abstract

Trait or phenotype prediction is one of pivotal tasks in the field of Genome-Wide Association Studies (GWAS). GWAS involves scanning genetic markers across genomes in order to find associations of genetic variants and human phenotypes. Despite numerous ongoing research in this field, understanding the genetic contribution to complex phenotypes still remains an open problem. In our paper, we propose a transformer model to perform quantitative phenotype prediction. We also propose a novel embedding method for the categorical data as a part of our transformer architecture, which can be utilized in other domains as well. Experimental results indicate that our model performs significantly better than baselines, on yeast dataset, for the traits that have considerably complex interactions.

1 Introduction

In light of recent advents in genetic sequencing techniques, the amount of available genomics data is rapidly increasing in the current decade [1]. Technological advancements in data generation from multiple levels of biological systems — including DNA sequence data [2], RNA expression levels [3], methylation patterns [4], other epigenetic markers [5], and proteomics [6] — have driven the field of translational bioinformatics in the past decade, producing huge chunks of data as researchers continually strive to develop complementary analysis tools [7]. This vast ocean of data can, in turn, help

*mohammad.erfan.mowlaei@temple.edu
†sanchari.biswas@temple.edu

in detection of genetic factors that contribute to diseases or improvement of breeding strategies in animals and plants. An important down-stream task in this venue is trait (phenotype) prediction. Traits are categorized as either qualitative (e.g., eye color) or quantitative (e.g., height). Our focus here is to predict quantitative traits using genotypes.

Models utilized for extracting key information out of genomic data, including, but not limited to, genotypes, require strong concepts and techniques of data mining. One underlying cause is that these genetic variants have linear or non-linear interactions, called additive and epistatic effects respectively, in determining trait values. Epistatic effects make phenotype prediction an exhausting task, since modeling such interactions demand significantly complex models compared to the additive effects. The other reason is the dimensionality of genomic data, where the cardinality of features is extremely larger than that of samples, referred to as the curse-of-dimensionality [8].

In this project, to address quantitative trait prediction problem, we present a novel deep learning model termed Split Transformer, utilizing transformer architecture, leveraging self-attention mechanism and embeddings, capable of capturing complex Single Nucleotide Polymorphism (SNP) interactions, or so called epistatic effects, towards phenotype changes. In the following, first, we will review related literature. Next, we present our methodology, followed by results and discussions.

2 Related Work

In practice, trait prediction is addressed using three different categories of approaches, namely Linear Midex Models (LMMs), machine learning (ML), and deep learning (DL) modeling. LMMs mostly assume that effects are additive and follow a normal distribution. Examples of such approaches are Best Linear Unbiased Prediction (BLUP) and several extended models including ridge regression BLUP (rrBLUP) [9], genomic relationship BLUP (GBLUP) [10], and single-step genomic BLUP (ssGBLUP) [11].

ML models generally struggle with curse-of-dimensionality and need careful feature selection. There are four major approaches to combat this problem using feature selection: filters, wrappers, embedded, and hybrid approaches. Filters, such as information gain [12] and Fisher score [13], depend majorly on statistical metrics in their estimation of correlation between features and

traits in order to select the best features for the phenotype. Compared with wrappers, these approaches are generally several times faster but deliver less optimal results. Wrappers utilize kernels of machine learning models to perform Sequential Feature Selection that can either be categorized as forward selection or backward elimination [14]. Embedded methods, on the other hand, use an internal mechanism based on techniques such as regularization or penalty to identify significant features during the training phase [15], such as Lasso [16], Ridge regression, and Elastic Nets [17]. Additionally, some embedded methods were specifically modified for applications in genomics. Some examples are Spike-and-Slab Lasso Generalized Linear Model (ssLasso GLMs) [18], Empirical Bayesian Elastic Net (EBEN) [19], its parallel version (parEBEN) [20], Multiple-trait Bayesian Lasso (MBL) [21], Bayesian Ridge Regression [22], and GPR [23]. The aforementioned methods specialize in trait prediction but they are designed for classification tasks. Thus, they cannot be applied to our problem, quantitative trait locus (QTL) prediction, where the phenotypes are quantitative traits. Hybrid methods depend on a combination of filters and wrappers in order to overcome the shortcomings of each of the former three and, in turn, deliver superior results [24, 25].

In recent years, DL models are widely used in down-stream genomics tasks, such as phenotype prediction. In [26], a denoising auto-encoder is implemented by first pre-training in an unsupervised manner and then utilizing for trait prediction. DeepGS [27] takes advantage of a deep Convolutional Neural Network (CNN) in order to learn underlying representation of genotype data for phenotype prediction. Another DL model based on dual-stream CNNs [28] is proposed for phenotype prediction, using saliency maps to assess an SNP's contribution to the phenotype.

Our proposed method falls into the DL category. We use a novel transformer model to tackle quantitative phenotype prediction. The rationale behind our choice of architecture is that self-attention used in transformers can take into account pair-wise feature interactions, requires zero feature engineering, and is currently state-of-the-art in many challenging tasks, such as computer vision and natural language processing.

3 Methodology

We start our work with our dataset, the Yeast dataset [29], and perform preprocessing. Then, we build our model, fine-tune its parameters, and perform

exploratory experiments, including investigating the contribution of our novel embedding method. In the end, we compare the performance of our model against baseline models to see how ours fares against them. We describe the process in more detail in the following sections. An overall diagram of our workflow is presented in Figure 1.

Figure 1: Workflow Block Diagram

3.1 Dataset Used

The dataset we have used in our work is the Yeast dataset [29]. This dataset contains sequenced genotype profiles of nearly 4,390 specimen, 28,220 attributes (SNPs), and 20 different quantitative growth traits. The samples in this dataset are a cross between a laboratory strain (BY) and an isolate form of vineyard (RM), encoded as -1 and 1, respectively. We use this dataset because here the phenotypes are quantitative traits and hence allows us to perform a quantitative trait locus (QTL) analysis.

3.2 Data Pre-processing

In our pre-processing section, among our general data cleaning measures, we apply min-max scaling to target traits in order to improve model performance, since DL models struggle, due to design of activation functions, when features and/or labels are not in range of [-1, 1]. Also known as min-max normalization, min-max scaling is a simple method that re-scales the data into decimals in range of [0, 1]. We also remove records with missing values for traits. The basic formula for a min-max scaling is given as following, where x is an original value, and x' is the normalized value.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

3.3 Proposed Model

Following the success of BERT [30] in natural language processing downstream tasks and wide usage of this architecture in computer vision tasks, such as Vision transformer [31] and Swin transformer [32], transformers gained a tremendous reputation and are applied to different domains, including genomics and proteomics. A revolutionary example is AlphaFold2 [33] that is used for protein structure prediction. Here, we present Split Transformer which comes with two novelties.

First, as the name implies, our model splits the features into chromosomes at the first level, and then to smaller windows at the second level, as depicted in Figure 2. In other terms, we form small windows, each containing a limited set of consecutive features that overlap with neighboring windows around the edges. Self-attention consumes quadratic memory with respect of data dimensionality. By splitting the features into windows, we trade off accurate global interactions for reduced attention memory overhead. To alleviate this phenomenon, we reduce the number of features at window level via max pooling, and apply another attention at chromosome level to capture nonlinear chromosome level interaction among the features at reduced resolution.

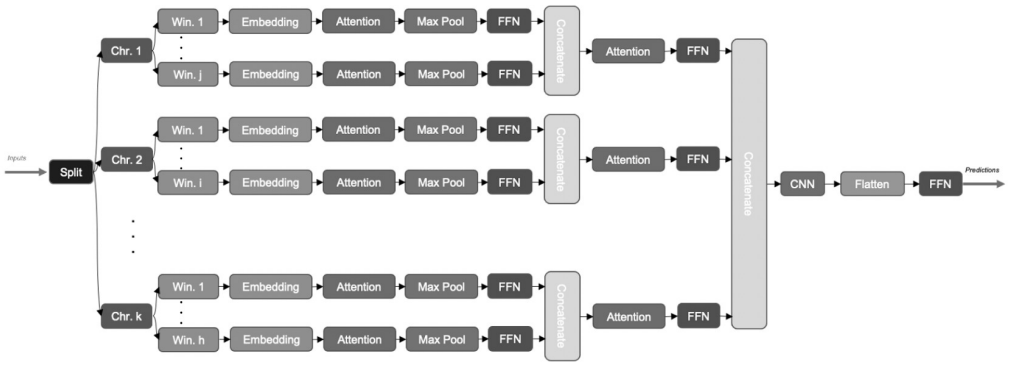

Figure 2: Split Transformer Architecture

Second, we introduce categorical embedding, an extension of positional embedding we came up with, which can encode categorical features and represent each category of each feature using a unique embedding vector. In natural language processing, we either have a word or we do not have it, thus we either include its embedding representation or we do not. However,

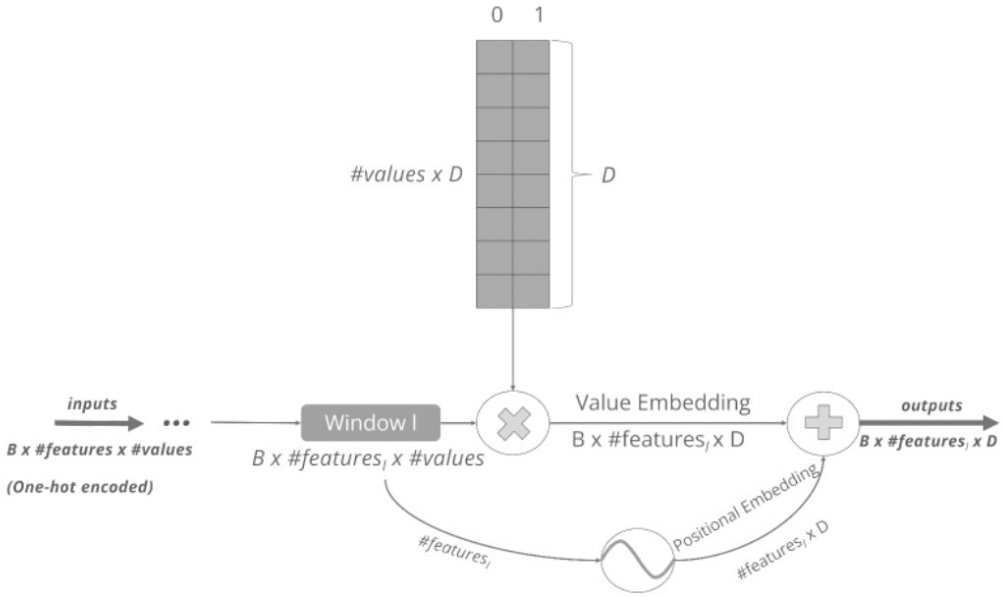

Figure 3: Categorical Embedding Workflow

in categorical feature cases, we always have to include respective representation of a feature, however, we need to distinguish between different value for it. For instance, if we are analyzing phones sold in 2022 based on features they have, such as battery capacity and their operating system, which are all categorically represented, we need to have different embedding vectors for 3500, 3700, 4000, 4500, and 5000 mA battery capacities and also different values for each of Android and iOS operating systems so that model can distinguish differences between the phones. In our case, each feature is binary, and we need two unique embedding vectors to represent different binary values for each feature. Therefore, in total, we need $2 \times 28220 = 56440$ unique embedding vectors to be able to represent all possible feature value combinations. In order to do that, we extended positional embedding used in transformers of natural language processing and computer vision tasks. The internal working mechanism of our categorical embedding is depicted in Figure 3, where B is the number of samples, $\#features_l$ is the number of features in *Window l* after splitting the chromosome into windows, $\#values$ is maximum number of categorical values that can be found in any feature in our window, hence 2 in our project, and D is embedding dimensionality, a hy-

perparameter of our model. In our case, we used 500, 50, and 32 for window size, window overlap size, and D parameter, respectively. We used Tensorflow [34] and Scikit-learn [35] Python packages and Google Colab platform to develop and run our models.

4 Results

In order to asses our proposed architecture, we performed extensive evaluations and benchmarked our model against fine-tuned baselines for the task, using 5 repetitions of random train-test split where 80% of the data was used for training the models and 20% was held out and used for evaluation of the model on unseen data. Fixed seeds were used for shuffling the data so that the same splits are acquired for all models in bechmarking. We examined 3 traits, namely *Cobalt Chloride*, *Copper Sulfate*, and *Diamide* having different ratios of epistatic to additive effects of quantitative trait loci (QTLs), according to the literature. Furthermore, we investigated the epistasis effect in our model performance and also the contribution of our proposed embedding is explored in a separate experiment. Our performance metric of choice is mean squared error (MSE), as it is widely used for regression tasks, such as ours.

4.1 Benchmarking

In order to evaluate how well our proposed model performs, we used 4 well-known and fine-tuned models as baselines. These model are Random Forest, Lasso, Elastic Net, and a Deep Neural Network (DNN) with three hidden fully connected layers of 14110, 7055, and 2822 neurons, respectively. For Random Forest, its implementation in R is used, where all hyperparameters are determined automatically from dimensions of the data. For Lasso and Elastic Net, their hyperparameters are tuned using a grid-search. As for our model, hyperparameters were fine tuned based on empirical studies we had on *Diamide*. As you can see in Figure 4, DNN is performing the worst among all. The underlying cause could be attributed to the fact that fully connected layers of DNN struggle with capturing local dependencies among the loci/features. Lasso is basically a Linear Regression (LR) model with a *L1* regularization term, and Elastic Net is an LR model with both *L1* and *L2* terms. We can see that Lasso and Elastic Net perform equally well as Random Forest on *Cobalt Chloride* and *Copper Sulfate*, but exceptionally

better on *Diamide*. As for Split Transformer, our proposed model, we observe improvement in performance, according to Equation (2), compared to the other models, on *Cobalt Chloride* and *Copper Sulfate* by 2% and 22%. respectively. However, our model performs slightly worse than Elastic Net and Lasso in case of *Diamide*.

$$Improvement = \frac{V_2 - V_1}{V_2} \tag{2}$$

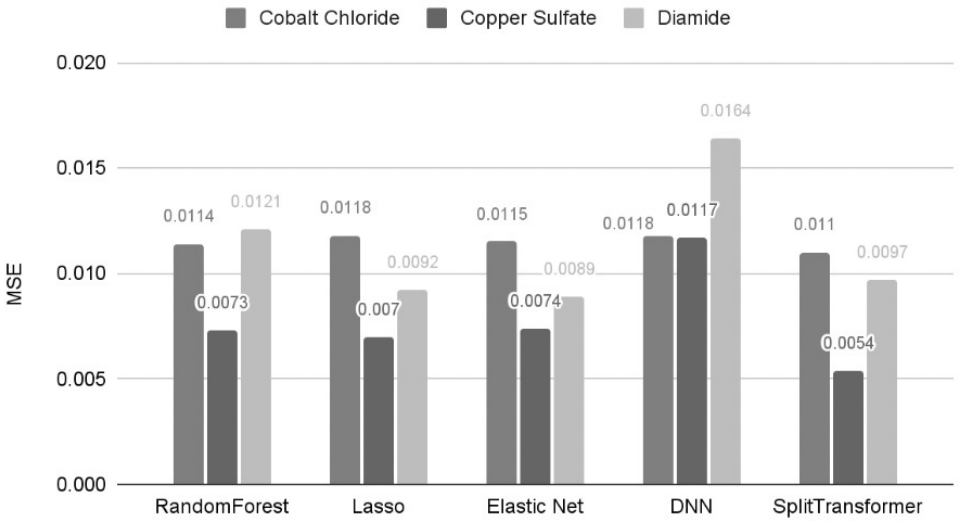

Figure 4: Performance Comparison based on MSE

To investigate the underlying cause for such difference in performance, we looked into supplementary materials of [29], where discovered quantitative trait loci were reported, listing additive and epistatic effects per quantitative trait. We summarized their findings in Table 1. Interestingly, in case of *Diamide*, we can see that almost all of the QTLs responsible for trait changes have additive effect, while the ratio of epistatic to additive traits for *Copper Sulfate* is relatively much higher. This entails that linear models, in this case Lasso and Elastic Net, can predict quantitative traits where almost all QTLs are additive quite accurately, while our proposed model shines where epistatic effects are involved to a larger scale and feature-feature interactions are relatively complex.

	#QTLs		
	Linear (Additive)	Non-Linear (Epistatic)	NL/L Ratio
Cobalt Chloride	33	9	0.27
Copper Sulfate	32	23	0.71
Diamide	56	6	0.11

Table 1: QTL contribution based on Bloom et al. (2014) findings

4.2 Embedding Study

As the last experiment, but not the least, we attempted to investigate the contribution of our categorical embedding layer in Split Transformer. To do so, we compared our final architecture to two different scenarios where the target for prediction was *Diamide*. In our final architecture, embeddings are placed inside each window, and we term it as local embedding. It entails that in this schema some features that are repeated in windows as a result of overlap between neighbouring windows will have two different embedding vectors, one per window, after the training is over. In an alternative scenario, called global embedding, instead of putting embedding inside the windows, we place them at the beginning of our model, entailing that overlapping features will still have the same embedding vector. Lastly, we removed embedding layer in each window, and replaced them with convolutional layers instead, in order to encode the input data. Similar to benchmarking, we repeated the experiment using 5 random splits with fixed seeds. The final results of this experiment are reported in Table 2, indicating that using embedding can improve performance compared to not using it, and also local embedding has a substantial impact on the performance of the model, compared to the two other scenarios. We speculate that learning several embeddings per feature in accordance with different features in locality of the SNP is the cause of such improvement.

	Compare to	
	ST - Global Embedding	ST - No Embedding
ST - Local Embedding	3.4%	4.3%
ST - Global Embedding	NA	0.8%

Table 2: Improvement comparison in ST model based on MSE performance of Diamide prediction

5 Conclusion

In this project, we tackled quantitative trait prediction task on yeast dataset using a novel transformer model, termed Split Transformer. In our empirical studies, we observed that our Split Transformer model is able to outperform other popular Machine Learning models, such as Random Forest, Lasso, Elastic Net, and DNN, in cases where predictions highly rely on complex feature interactions. We also observe that our proposed categorical embedding makes it possible to use embedding for any categorical input data. In such an instance, local (categorical) embedding leads to better performance compared to global (categorical) embedding or to not using any embedding at all. We continue working on improving and refining our model and our future work involves improving the model to better predict the traits where the effects are mostly additive.

6 Acknowledgements

We would like to thank Prof. Pei Wang for continuous support on the project. We also thank Prof. Xinghua Shi for her valuable advice regarding dataset selection, phenotype prediction, and feedbacks on model development.

References

[1] A. Alemany, M. Florescu, C. S. Baron, J. Peterson-Maduro, and A. Van Oudenaarden, "Whole-organism clone tracing using single-cell sequencing," *Nature*, vol. 556, no. 7699, pp. 108–112, 2018.

[2] M. L. Metzker, "Sequencing technologies—the next generation," *Nature reviews genetics*, vol. 11, no. 1, pp. 31–46, 2010.

[3] F. Ozsolak and P. M. Milos, "Rna sequencing: advances, challenges and opportunities," *Nature reviews genetics*, vol. 12, no. 2, pp. 87–98, 2011.

[4] P. W. Laird, "Principles and challenges of genome-wide dna methylation analysis," *Nature Reviews Genetics*, vol. 11, no. 3, pp. 191–203, 2010.

[5] P. J. Park, "Chip–seq: advantages and challenges of a maturing technology," *Nature reviews genetics*, vol. 10, no. 10, pp. 669–680, 2009.

[6] A. Altelaar, J. Munoz, and A. J. Heck, "Next-generation proteomics: towards an integrative view of proteome dynamics," *Nature Reviews Genetics*, vol. 14, no. 1, pp. 35–48, 2013.

[7] M. D. Ritchie, E. R. Holzinger, R. Li, S. A. Pendergrass, and D. Kim, "Methods of integrating data to uncover genotype–phenotype interactions," *Nature Reviews Genetics*, vol. 16, no. 2, pp. 85–97, 2015.

[8] R. E. Bellman, *Dynamic programming and stochastic control processes / by Richard Bellman (ASTIA Document no).* Rand Corp, 2022.

[9] B. Hayes, M. Goddard *et al.*, "Prediction of total genetic value using genome-wide dense marker maps," *Genetics*, vol. 157, no. 4, pp. 1819–1829, 2001.

[10] P. M. VanRaden, "Efficient methods to compute genomic predictions," *Journal of dairy science*, vol. 91, no. 11, pp. 4414–4423, 2008.

[11] A. Legarra, I. Aguilar, and I. Misztal, "A relationship matrix including full pedigree and genomic information," *Journal of dairy science*, vol. 92, no. 9, pp. 4656–4663, 2009.

[12] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[13] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," *Advances in neural information processing systems*, vol. 18, 2005.

[14] K. Yan, L. Ma, Y. Dai, W. Shen, Z. Ji, and D. Xie, "Cost-sensitive and sequential feature selection for chiller fault detection and diagnosis," *International Journal of Refrigeration*, vol. 86, pp. 401–409, 2018.

[15] H. Wang, X. Jing, and B. Niu, "A discrete bacterial algorithm for feature selection in classification of microarray gene expression cancer data," *Knowledge-Based Systems*, vol. 126, pp. 8–19, 2017.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[17] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[18] Z. Tang, Y. Shen, X. Zhang, and N. Yi, "The spike-and-slab lasso generalized linear models for prediction and associated genes detection," *Genetics*, vol. 205, no. 1, pp. 77–88, 2017.

[19] A. Huang, S. Xu, and X. Cai, "Empirical bayesian elastic net for multiple quantitative trait locus mapping," *Heredity*, vol. 114, no. 1, pp. 107–115, 2015.

[20] J. Wen, C. T. Ford, D. Janies, and X. Shi, "A parallelized strategy for epistasis analysis based on empirical bayesian elastic net models," *Bioinformatics*, vol. 36, no. 12, pp. 3803–3810, 2020.

[21] D. Gianola and R. L. Fernando, "A multiple-trait bayesian lasso for genome-enabled analysis and prediction of complex traits," *Genetics*, vol. 214, no. 2, pp. 305–331, 2020.

[22] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[23] O. A. Montesinos-López, J. Crossa, E. P. Setiawan, and D. U. Wutsqa, "Prediction of count phenotypes using high-resolution images and genomic data," *G3*, vol. 11, no. 2, p. jkab035, 2021.

[24] R. Alanni, J. Hou, H. Azzawi, and Y. Xiang, "A novel gene selection algorithm for cancer classification using microarray datasets," *BMC medical genomics*, vol. 12, no. 1, pp. 1–12, 2019.

[25] M. K. Ebrahimpour, H. Nezamabadi-Pour, and M. Eftekhari, "Ccfs: A cooperating coevolution technique for large scale feature selection on microarray datasets," *Computational biology and chemistry*, vol. 73, pp. 171–178, 2018.

[26] J. Chen and X. Shi, "A sparse convolutional predictor with denoising autoencoders for phenotype prediction," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2019, pp. 217–222.

[27] W. Ma, Z. Qiu, J. Song, J. Li, Q. Cheng, J. Zhai, and C. Ma, "A deep convolutional neural network approach for predicting phenotypes from genotypes," *Planta*, vol. 248, no. 5, pp. 1307–1318, 2018.

[28] Y. Liu, D. Wang, F. He, J. Wang, T. Joshi, and D. Xu, "Phenotype prediction and genome-wide association study using deep convolutional neural network of soybean," *Frontiers in genetics*, vol. 10, p. 1091, 2019.

[29] J. S. Bloom, I. Kotenko, M. J. Sadhu, S. Treusch, F. W. Albert, and L. Kruglyak, "Genetic interactions contribute less than additive effects to quantitative trait variation in yeast," *Nature communications*, vol. 6, no. 1, pp. 1–6, 2015.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[32] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[33] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems,"

2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.