This Is My Title
Along With More Clarification

by

Anthony J La Barca

Submitted in Partial Fulfillment of the

Requirements for the Degree

Bachelor of Science in Physics and Astronomy

Supervised by Segev BenZvi, Ph.D.

Department of Physics and Astronomy
School of Arts and Sciences

University of Rochester

Rochester, New York

2023

For someone a long time ago, in a galaxy far, far away...

Acknowledgments

Consider adding acknowledgements.

Abstract

This is an abstract! Read the paper :)

Contents

Acknowledgments **iii**

Abstract **iv**

List of Tables **vii**

List of Figures **ix**

1 Introduction **1**

1.1 Citation styles 1

2 Deep Learning Techniques **2**

2.1 Convolutional Neural Networks 4

2.1.1 CNNs on Spectroscopic Data 5

2.2 Introduction of Transformers 7

2.2.1 ViT on Spectroscopic Data 8

3 Creation and Training of Spectral ViT **9**

3.1 Creation of Spectral ViT 9

3.1.1 Validation of Spectral ViT Architecture 12

3.2 Creation of Synthetic DESI Spectra 14
3.2.1 PreProcessing of DESI Spectra 15
3.3 Training of Neural Networks 15
3.3.1 CNN Training 15
3.3.2 Transformer Training 16

4 Results 20

4.1 V1 20
4.2 V2 24
4.2.1 Epoch 42 24
4.2.2 Epoch49 26
4.3 Binary Classification 28
4.3.1 Epoch49 30

A Detailed Diagrams of Various Neural Networks 34

A.1 Multi-Layer Perceptron (Fully Connected Feed-Forward Networks) . . 34
A.2 Convolutional Neural Networks 34
A.3 Transformers 34
A.3.1 Spectral ViT 34
A.3.2 Multi-Head Attention 34

B Title of Appendix B 35

List of Tables

3.1 This is a table 18
3.2 This is a table 18
3.3 This is a table 18

List of Figures

2.1 A simple MLP with three hidden layers. Each node is connected to every node in the previous and next layer. The lines represent the weights, and each node has an associated bias. Figure adapted from Neutelings (2021). 3
2.2 CNN Diagnostics: ROC Curve (left) and Confusion Matrix (right) . . 5
2.3 Max value of the output vector for the CNN. 6
2.4 CNN Diagnostics 7

3.1 Visual representation of the embeddings for a sample of spectra split into 100 patches of length 300. 11

3.2 Synthetic spectra used to verify performance of SpectralViT 13

3.3 Synthetic spectra used to verify performance of SpectralViT 14

3.4 Caption 16

3.5 Training of V1 17

3.6 Training of V1 Big 17

3.7 Training of V2 19

4.1 V1 - No cut 21

4.2 Max Y 22

4.3 V1 - 0.9999 23

4.4 V2 - 42 - No cut 24

4.5 V2 - 42 - Max pred 24

4.6 V2 -42 - 0.99999 25

4.7 V2 - 42- 0.99999 25

4.8 V2 - 49 - No cut 26

4.9 V2 - 49 - maxy 26

4.10 V2 - 49 - .99 27

4.11 V2 - 49 - 0.99999 27

4.12 V2 - 42 - binary - no cut 28

4.13 V2 - 42 - binary - maxy 28

4.14 V2 - 42 - binary - .99 29

4.15 V2 - 42 - binary - .99999997 29

4.16 V2 - 49 - binary - no cut 30

4.17 V2 - 49 - binary - maxy 30

4.18 V2 - 49 - binary - .99 31
4.19 V2 - 49 - binary - .99999997 31

1. Introduction

1.1 Citation styles

These are the different citation styles for author-year.

The standard `\cite` command produces the following output: Cybenko 1989.

The `\textcite` command produces the following output: Cybenko (1989).

The `\parencite` command produces the following output: (Cybenko 1989).

The `\footcite` command produces a footnote citation[1].

[1] Cybenko 1989.

2. Deep Learning Techniques

To aid in the classification of supernovae, deep learning techniques can provide flexibility and scalability that have been previously unattainable by manual methods. The simplest and most widely used architecture in deep learning are the feed-forward fully connected neural networks, or multi-layer perceptrons (MLPs) (Popescu et al. 2009). These networks are composed of series of “layers” of neurons, or nodes, that are “fully connected” to the previous layer (Figure 2.1). Given an input vector $\vec{x}$ with n values, an MLP applies a series of weights to each element, resulting in a linear transformation from $\vec{x}$ to $\vec{y}$, where $\vec{y}$ is a vector of length m. This transformation is then followed by a bias added to each element of $\vec{y}$, and an activation function applied to each element. This can be represented mathematically as

$$\vec{y} = \sigma(\mathbf{W}\vec{x} + \vec{b}) \tag{2.1}$$

where $\mathbf{W}$ is an $m \times n$ matrix of weights, $\vec{b}$ is a vector of length m representing the bias, and σ is the activation function. The activation function can be any function, but traditionally they are chosen to be continuous, non-linear, monotonically increasing, and differentiable. The most common activation functions are the sigmoid function,

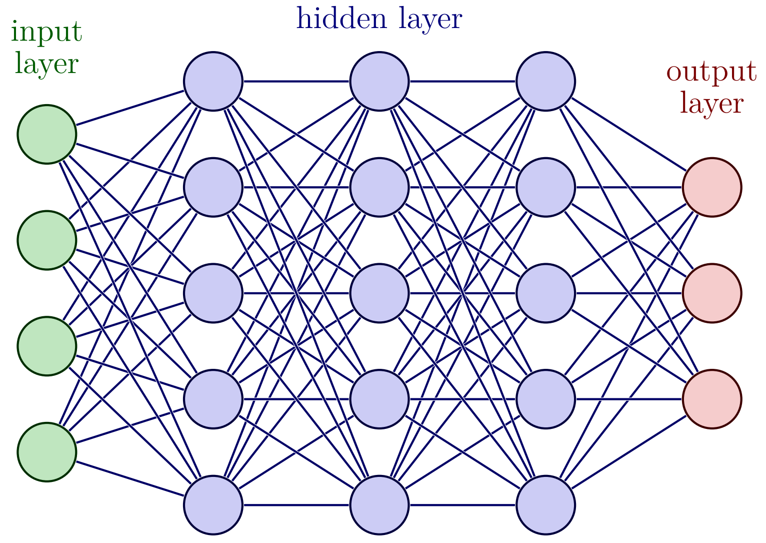


Figure 2.1: A simple MLP with three hidden layers. Each node is connected to every node in the previous and next layer. The lines represent the weights, and each node has an associated bias. Figure adapted from Neutelings (2021).

$\sigma(x) = \frac{1}{1+e^{-x}}$, hyperbolic tangent, $\sigma(x) = \tanh(x)$, and the rectified linear unit, or ReLU, $\sigma(x) = \max(0, x)$.

The output layer of the network is another vector tailored for the particular purpose of the MLP (i.e. for classification, the output is a vector of probabilities that the input data belongs to, see Appendix A.1).

The MLP architecture, while simplistic, is very powerful. Cybenko (1989) showed that for any boolean function (i.e. 2 class classification), a MLP with a single hidden layer could approximate any function arbitrarily well; this is known as the universal approximation theorem. The theorem is the basis for the success of MLPs in many fields, but it is also the reason that MLPs are not the best choice for all problems. The main issue with MLPs is that they are not invariant to translations, which is essential for many problems including image recognition, time series analysis, and natural language processing. **Therefore, alternative options have been proposed** ****

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) were originally developed as a solution to positional differences in the input data (Fukushima 1979). This architecture was shown to be effective in image recognition tasks by LeCun, Huang, and Bottou (2004) and then popularized after the success of AlexNet in the ImageNet challenge Krizhevsky, Sutskever, and Hinton (2012). The networks have since been applied to many other vision tasks and various fields, such as NLP and audio processing. CNNs are composed of a series of convolutional layers, usually followed by a pooling layer and a fully connected layer (Appendix A.2). The convolutional layers are composed of a series of convolutional filters, which are commonly referred to as feature maps. The convolutional filters are applied across the entire input space giving the CNN its invariance to translations.

2.1.1 CNNs on Spectroscopic Data

CNNs excel at identifying patterns throughout their input space, stemming from their spacial invariance. Theoretically, this should translate to spectral classification quite well. In fact, CNNs have been applied to spectroscopic data in the past, including on the DESI dataset. Parks et al. (2018) used a CNN to detect strong emission lines in DESI spectra with great success. * Talk about 1D CNN currently running, and preprocessing used * figures/cnn/cnn_rocfull.png

figures/cnn/cnn_cmfull.png

Figure 2.2: CNN Diagnostics: ROC Curve (left) and Confusion Matrix (right)

This work provides a baseline for the use of CNNs on supernovae classification, **but leaves more to be desired.** Therefore, an alternate architecture was proposed by * Cite Eddies work * which augments the preprocessing of the spectra with a conversion to a 2D image. This 2D image was then fed to a more traditional vision CNN architecture (Appendix A.2). This architecture was shown to train quickly (less than 1 hour on a single GPU), but it was not able to achieve astonishingly high accuracy. Figure 2.2 shows the ROC curve and confusion matrix for the CNN trained on synthetic spectra.

The maximum value of the output vector was used to determine the predicted class,

figures/cnn/cnn_max_ypred.png

Figure 2.3: Max value of the output vector for the CNN.

figures/cnn/cnn_roc99.png

(a) ROC curve

figures/cnn/cnn_cm99.png

(b) Confusion Matrix

Figure 2.4: CNN Diagnostics

which may not be the best choice for this problem. why not? Figure 2.3 shows the distribution of the maximum value found. As shown, there are approximately 20 thousand spectra that are classified very confidently, but there are many more that are not.

A **much better ROC curve and confusion matrix** are produced by evaluating the diagnostics for only highly confident classifications (Figure 2.4). Therefore, it is clear that the CNN, when confident in its classification, produces accurate results. This cut, however, is not ideal, removing *insert percent*% of the data.

The next step in training would be to either increase the confidence of the CNN or move to a different architecture, with the hopes of increasing not only the overall accuracy of the networks, but also the number of confident classifications.

2.2 Introduction of Transformers

Transformers are a relatively new architecture introduced in 2017 by Vaswani et al. (2017) for natural language processing (NLP) tasks. The original architecture consists of an encoder-decoder system. The encoder accepts a series of tokens and produces a series of vectors representing the input data via a series of self-attention layers and feed-forward layers. The decoder then takes the output of the encoder, and produces a series of tokens, one for each input token.

Once transformers were shown to have remarkable success in NLP tasks, they were quickly adapted to other fields, such as vision. Dosovitskiy et al. (2020) developed a vision transformer (ViT) architecture that differed from the original transformer encoder by replacing the tokenized input with a more involved preprocessing step. In short, the input image is broken into a series of patches, which are then flattened into a vector. These vectors, along with positional encodings, are then fed into the transformer architecture. For classification tasks, the first input token is replaced with a class token. After passing through the transformer, the class token is then run through a fully connected layer to produce the final probabilities.

2.2.1 ViT on Spectroscopic Data

Previous implementations of transformers have been shown to have characteristics beneficial to the classification of spectral data. A transformer's ability to learn contextual information is essential in spectral classification. A broad absorption line, for example, may be indicative of a Type Ia supernova if in one part of the spectrum, but may be indicative of a Type II supernova if in another part of the spectrum. This contextual information is not easily learned by a CNN, as the convolutional layers are not able to learn the importance of certain parts of the input space. Attention can also play a role in identifying the purpose of features that are not in a standard location. For example, a **non-k corrected s**pectrum might have a continuum pattern at different locations in the spectrum, but the overall shape would be similar. This change in sizing would be difficult to learn with fixed filters in a CNN, but would be identified based on their positional importance by a transformer. In addition to this, ViTs have been shown to outperform CNNs on vision tasks, which shows they are capable of focusing on learned features.

* Include caveat about the fact that ViTs take longer to train than CNNs *

3. Creation and Training of Spectral ViT

In order to examine the effectiveness of the transformer architecture for supernovae spectral classification, a series of steps were taken. First, an in-house transformer architecture was created based on the traditional ViT architecture Dosovitskiy et al. 2020, which was quickly found to be trainable on a synthetic dataset. Next, a synthetic dataset using DESI spectra was created under a variety of preprocessing conditions. The previously created CNN and new transformer architectures were trained on each variation of preprocessed data. Finally, these training sessions were then used to determine the optimal training conditions for non redshift corrected data.

3.1 Creation of Spectral ViT

The Spectral ViT architecture was created to be a direct extension of the ViT architecture Dosovitskiy et al. 2020 to be used for spectral classification, coded in the `PyTorch` deep learning framework. The Spectral ViT architecture is shown graphically in Appendix A.3.1. The Spectral ViT architecture is composed of three main components: the pre-processor, the encoder, and the classifier.

The pre-processing component is responsible for taking the (already preprocessed) input spectra and converting it into a series of vectors that the transformer can interpret. Considering a group of N spectra, each with 10000 pixels, the each group is split into a set number of patches (approximately 100). Each patch is then linearly mapped via a fully connected network to a vector three times the patch size. The sample is now a set of of size $N \times 101 \times 300$. A classification token of the sample dimensionality as the patches is then added to the beginning of each sample, initialized randomly. In order for the transformer to properly understand the positional relationship between each patch, an embedding is added to each patch. This embedding is a scalar function based on the size of the patch is calculated as follows:

$$\text{Embedding}_{ij} = \begin{cases} \sin\left(\frac{i}{10000^{(j/\text{patch size})}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{((j-1)/\text{patch size})}}\right) & \text{if } j \text{ is odd} \end{cases}, \tag{3.1}$$

where i is the position of the patch in the vector, and j is the position of the patch in the sample Vaswani et al. 2017. A visual representation of the embeddings for the sample is shown in Fig. 3.1. Once the embeddings are added element-wise to the patches, the resulting tensor (of size $N \times 101 \times 300$) is then passed through the encoder.

`figures/embeddings.png`

Figure 3.1: Visual representation of the embeddings for a sample of spectra split into 100 patches of length 300.

The encoder is the main component of any ViT architecture, as it contains the multi-head attention and feed-forward layers. The encoder is composed of a set number of transformer blocks, each of which contains layer normalization, multi-head attention, another layer normalization, and finally a feed-forward layer. The multi-head attention layer is a scaled dot-product attention layer found in Vaswani et al. (2017). Each patch is passed through three separate linear layers, each with a different set of weights, resulting in three sets of vectors: the query (Q), key (K), and value (V).

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \tag{3.2}$$

The dot-product between the query and key vectors is then calculated, scaled by the square root of the dimensionality of the query vector, and then passed through a softmax function. The resulting attention weights are then multiplied by the value vectors (Equation 3.2). This is simultaneously done for each head in the multi-head attention layer. Each result is then concatenated together, and then passed through a linear layer to reduce the dimensionality to the size of the query vector. This resulting vector is then added element-wise to the original patches, normalized, and then passed through a MLP, resulting in a tensor of size $N \times 101 \times 300$. This again is added element-wise to the original patches. Each step is repeated for a set number of transformer blocks, resulting in a tensor of size $N \times 101 \times 300$.

The final component of the Spectral ViT architecture is the classifier. The classifier takes in only the first patch of each sample, which has been designated as the classification token. The classification token is passed through a linear layer to reduce the dimensionality to the number of classes, and then passed through a softmax function to produce a probability distribution over the classes. Therefore, the resulting tensor is of size $N \times 1 \times 6$, for our 6 classifications of supernovae.

3.1.1 Validation of Spectral ViT Architecture

After the creation of the Spectral ViT architecture, ability to train effectively was tested. A synthetic dataset, consisting of a consistent continuum with Gaussian peaks placed at predetermined locations, was created. Certain combinations of peak locations were chosen to represent an arbitrary 'class' of supernovae. These peaks, our synthetic emission lines, were given random amplitudes and widths, simulating variability, with a maximum allowed value. This maximum allowed value was then used to add Gaussian noise to the signal: either 2 or 10 times the signal to simulate noisy ($S/N = 2$), or very good data ($S/N = 10$). Examples of the synthetic spectra with different continuum profiles are shown in Fig. 4.19.

These datasets with a signal-to-noise of 10 were trivially separable by a smaller Spectral ViT architecture, which was able to achieve 100% accuracy on the testing set. The lower quality data, however, was more difficult to separate, only achieving an 33.975% test accuracy.

figures/pre_testing/SN2_example.png

(a) SNR = 2

figures/pre_testing/SN10_example.png

(b) SNR=10

Figure 3.2: Synthetic spectra used to verify performance of SpectralViT

figures/pre_testing/SNR2_training_epoch.png

(a) SNR = 2

figures/pre_testing/SNR10_training_epoch.png

(b) SNR=10

Figure 3.3: Synthetic spectra used to verify performance of SpectralViT

3.2 Creation of Synthetic DESI Spectra

Once the Spectral ViT architecture was shown to be able to train effectively on synthetic data, the architecture was ready to be trained on DESI data. In order to create a large enough training set, authentic DESI spectra were used as a base to create synthetic spectra.

** Talk to BenZvi about what to put in this section / who to cite for all of the work **

3.2.1 PreProcessing of DESI Spectra

Once the spectra's were created and saved as DESI files, they needed to be extracted, preprocessed, split into training, testing, and validation sets, only then could they be saved, and used to train the Spectral ViT architecture. The preprocessing method developed by ** Cite eddies paper **, and is split into 3 main steps: z correction, rebinning / down sampling, and normalization.

The Z correction step was used to move the spectra back into the rest frame using the redshift fitted to the original spectra by the DESI pipeline. Next, all artifacts in the spectra (masks, bad pixels, etc.) were removed. The spectra were then re-binned and downsampled to a variable resolution (default 3600). Afterwards, the spectra were normalized by moving the max and min values to 1 and 0 respectively. Finally, for the CNN datasets, the spectra were split into a 2D array of equal height and width. After this, the spectra were split into training, testing and validation datasets that comprised of 60%, 20%, and 20% of the total dataset, respectively.

3.3 Training of Neural Networks

3.3.1 CNN Training

CNN training was conducted using DESI spectra downsampled to a resolution of 3600, and put into a 2D array of equal height and width. The CNN architecture was developed by ** Cite eddies paper **, and has properties shown in Table 3.1. This CNN was trained for a maximum of 50 epochs, with a batch size of 50, and non-variable hyperparameters (Table 3.2). The timeline of the training of the CNN figures/blackbox.jpeg

Figure 3.4: Caption

architecture is shown in Fig. 3.4.

3.3.2 Transformer Training

The Spectral ViT architecture, idealistically, should have the capability to be trained on non rest-frame corrected spectra. In an effort to test how down-sampling had affected the training of the Spectral ViT architecture, the Spectral ViT architecture was trained on various downsampling values. Based on these models, a downsampling value was chosen for non-rest-frame corrected spectra. The Spectral ViT architecture was trained on DESI spectra downsampled to a resolution of 3600, 1800, and 900. The Spectral ViT architecture was trained for a maximum of 100 epochs, with batch sizes of 50, and non-variable hyperparameters (Table 3.3).

figures/v1/vit_model_V1_original_redotraining_after.png

Figure 3.5: Training of V1

figures/v1/vit_model_V1_bigtraining_after.png

Figure 3.6: Training of V1 Big

Spam	**Ni**	**Swallow**	**Shrubbery**
A	1	2	3
E	3	4	5
C	6	9	3
M	4	1	1

Table 3.1: This is a table

Spam	**Ni**	**Swallow**	**Shrubbery**
A	1	2	3
E	3	4	5
C	6	9	3
M	4	1	1

Table 3.2: This is a table

Spam	**Ni**	**Swallow**	**Shrubbery**
A	1	2	3
E	3	4	5
C	6	9	3
M	4	1	1

Table 3.3: This is a table

`figures/v2/vit_model_V2training_after.png`

Figure 3.7: Training of V2

4. Results

4.1 V1

`figures/v1/rocfull.png`

(a) ROC

`figures/v1/cmfull.png`

(b) CM

Figure 4.1: V1 - No cut

figures/v1/max_ypred.png

Figure 4.2: Max Y

figures/v1/roc99999.png

(a) ROC

figures/v1/cm99999.png

(b) CM

Figure 4.3: V1 - 0.9999

4.2 V2

4.2.1 Epoch 42

figures/v2/vit_model_V2rocfulle_e42.png

(a) ROC

figures/v2/vit_model_V2cmfull_e42.png

(b) CM

Figure 4.4: V2 - 42 - No cut

figures/v2/vit_model_V2max_ypred_42.png

Figure 4.5: V2 - 42 - Max pred

figures/v2/vit_model_V2roc99_e42.png

(a) ROC

figures/v2/vit_model_V2cm99_e42.png

(b) CM

Figure 4.6: V2 -42 - 0.99999

figures/v2/vit_model_V2roc99999_e42.png

(a) ROC

figures/v2/vit_model_V2cm99999_e42.png

(b) CM

Figure 4.7: V2 - 42- 0.99999

4.2.2 Epoch49

figures/v2/vit_model_V2rocfulle_e49.png

figures/v2/vit_model_V2cmfull_e49.png

(a) ROC

(b) CM

Figure 4.8: V2 - 49 - No cut

figures/v2/vit_model_V2max_ypred_49.png

Figure 4.9: V2 - 49 - maxy

figures/v2/vit_model_V2roc99_e49.png

(a) ROC

figures/v2/vit_model_V2cm99_e49.png

(b) CM

Figure 4.10: V2 - 49 - .99

figures/v2/vit_model_V2roc99999_e49.png

(a) ROC

figures/v2/vit_model_V2cm99999_e49.png

(b) CM

Figure 4.11: V2 - 49 - 0.99999

4.3 Binary Classification

figures/v2/vit_model_V2rocfull_binary_e42.png

figures/v2/vit_model_V2cmfull_binary_e42.png

(a) ROC **(b)** CM

Figure 4.12: V2 - 42 - binary - no cut

figures/v2/vit_model_V2max_ypred_binary_42.png

Figure 4.13: V2 - 42 - binary - maxy

figures/v2/vit_model_V2roc99_binary_e42.png

(a) ROC

figures/v2/vit_model_V2cm99_binary_e42.png

(b) CM

Figure 4.14: V2 - 42 - binary - .99

figures/v2/vit_model_V2roc99999997_binary_e42.png

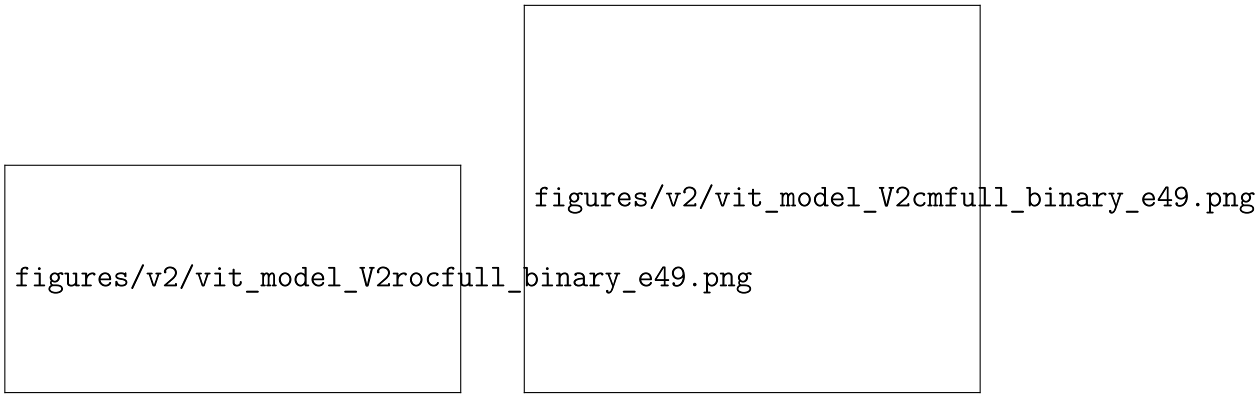
(a) ROC

figures/v2/vit_model_V2cm99999997_binary_e42.pn

(b) CM

Figure 4.15: V2 - 42 - binary - .99999997

4.3.1 Epoch49


figures/v2/vit_model_V2rocfull_binary_e49.png
figures/v2/vit_model_V2cmfull_binary_e49.png

(a) ROC **(b)** CM

Figure 4.16: V2 - 49 - binary - no cut

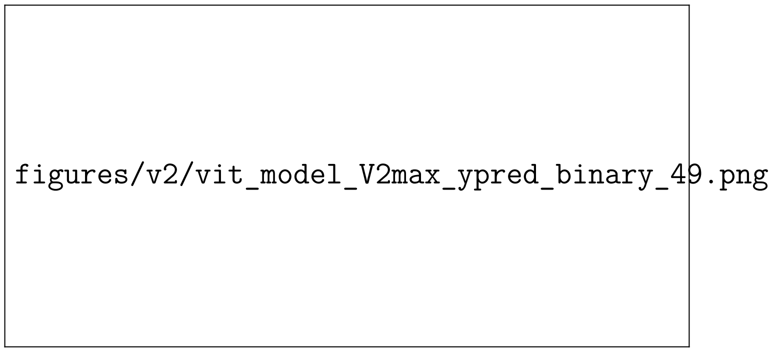


Figure 4.17: V2 - 49 - binary - maxy

figures/v2/vit_model_V2roc99_binary_e49.png

(a) ROC

figures/v2/vit_model_V2cm99_binary_e49.png

(b) CM

Figure 4.18: V2 - 49 - binary - .99

figures/v2/vit_model_V2roc99999997_binary_e49.png

(a) ROC

figures/v2/vit_model_V2cm99999997_binary_e49.pn

(b) CM

Figure 4.19: V2 - 49 - binary - .99999997

Bibliography

Cybenko, G. (Dec. 1, 1989). “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4, pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

Dosovitskiy, Alexey et al. (2020). “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *CoRR* abs/2010.11929. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

Fukushima, Kunihiko (1979). “Self-Organization of a Neural Network Which Gives Position-Invariant Response”. In: *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI’79. Tokyo, Japan: Morgan Kaufmann Publishers Inc., pp. 291–293. ISBN: 0934613478. DOI: 10.5555/1624861.1624928.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

LeCun, Y., Fu Jie Huang, and L. Bottou (2004). “Learning methods for generic object recognition with invariance to pose and lighting”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2, II–104 Vol.2. DOI: 10.1109/CVPR.2004.1315150.

Neutelings, Izaak (Sept. 26, 2021). *Neural networks*. URL: https://tikz.net/neural_networks/.

Parks, David et al. (Jan. 2018). “Deep learning of quasar spectra to discover and characterize damped Lyα systems”. In: *Monthly Notices of the Royal Astronomical Society* 476.1, pp. 1151–1168. ISSN: 0035-8711. DOI: 10.1093/mnras/sty196. eprint: https://academic.oup.com/mnras/article-pdf/476/1/1151/24261051/sty196.pdf. URL: https://doi.org/10.1093/mnras/sty196.

Popescu, Marius-Constantin et al. (July 2009). “Multilayer perceptron and neural networks”. In: *WSEAS Transactions on Circuits and Systems* 8.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

A. Detailed Diagrams of Various Neural Networks

A.1 Multi-Layer Perceptron (Fully Connected Feed-Forward Networks)

A.2 Convolutional Neural Networks

A.3 Transformers

A.3.1 Spectral ViT

A.3.2 Multi-Head Attention

B. Title of Appendix B