Adoption of the Transformer Architecture for Transient Detection in the DESI Survey

by

Anthony J La Barca

Submitted in Partial Fulfillment of the

Requirements for the Degree

Bachelor of Science in Physics and Astronomy

Supervised by Professor Segev BenZvi

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 Supernova Classification 1

1.2 Supernova Identification 3

1.3 Dark Energy Spectroscopic Instrument 3

1.4 Supernova Classification with Machine Learning 4

1.5 Problem Statement 4

2 Deep Learning Techniques **6**

2.1 Multi-Layer Perceptrons 6

2.2 Convolutional Neural Networks 8

2.2.1 CNNs on Spectroscopic Data in Astronomy 10

2.3 From CNNs to Transformers 13

2.3.1 ViT on Spectroscopic Data 14

3 Creation and Training of Spectral ViT 16

3.1 Creation of Spectral ViT 16

3.1.1 Validation of Spectral ViT Architecture 19

3.2 Creation of Synthetic DESI Spectra 20

3.2.1 Pre-Processing of DESI Spectra 21

3.3 Training of Neural Networks 23

3.3.1 CNN Training 23

3.3.2 Transformer Training 23

4 Spectral ViT Model Performance 27

4.1 V1 27

4.2 V2 29

4.2.1 Binary Classification 31

A Detailed Diagrams of Various Neural Networks 36

A.1 Multi-Layer Perceptron (Fully Connected Feed-Forward Networks) . . 36

A.2 Convolutional Neural Networks 36

A.3 Transformers 36

A.3.1 Spectral ViT 36

A.3.2 Multi-Head Attention 36

B Title of Appendix B 38

List of Tables

3.1 Hyperparameters of the CNN used to classify DESI spectra. CNN adapted from Sepeku (2022). 23
3.2 Hyperparameters of the smaller and larger Spectral ViT architecture used to classify DESI spectra. 24

A.1 Architecture of the CNN used to classify DESI spectra. CNN adapted from Sepeku (2022). 37

List of Figures

1.1 Supernova Classification 2

2.1 Diagram of a Multi-Layered Perceptron 7
2.2 Example of a CNN Unit 9
2.3 Traditional CNN Architecture for Classification 10
2.4 CNN diagnostics 11
2.5 CNN's Confidence in Classification 12
2.6 CNN diagnostics with a confidence cut 13
2.7 Transformer Architecture 14
2.8 ViT Architecture 15

3.1 Visual representation of the embeddings for a sample of spectra. The x-axis represents the index of the patch, the y-axis represents the index of the hidden dimension, and the color represents the value of the embedding. 18

3.2 Synthetic spectra used to verify performance of SpectralViT. Red and green vertical lines represent features used to seperate the data into disticnt classes for SNR=2 and SNR=10, respectively. Two classes are shown for each SNR. 20
3.3 Training of Spectral ViT on synthetic datasets. The model was able to reach 100% accuracy on the test set of SNR=10 (top), but failed to reach an accuracy of 50% on the test set of SNR=2 (bottom). 21
3.4 Training of the CNN on synthetic DESI spectra. 22
3.5 Training of the small architecture on redshift-corrected spectra downsampled to 3600 bins. Over-fitting was determined to have occurred by Epoch 31. 24
3.6 Training of the large architecture on Redshift-corrected spectra downsampled to 3600 bins. No convergence above random guessing was observed. 25
3.7 Training of the small architecture on Redshift-corrected spectra downsampled to 1800 bins. No convergence above random guessing was observed. 25
3.8 Training of the small architecture on Redshift-corrected spectra downsampled to 900 bins. No convergence above random guessing was observed. 25
3.9 Training of the small architecture on non Redshift-corrected data downsampled to 3600 bins. Over fitting was determined to have occurred by Epoch 26. 26

4.1 Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right) 27
4.2 Max value of the output vector from the Spectral ViT V1. 28
4.3 Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99% confidence cut 28
4.4 Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99.9999% confidence cut 29
4.5 Spectral ViT V2 Diagnostics: ROC Curve (left) and Confusion Matrix (right) 29
4.6 Max value of the output vector from the Spectral ViT V2. 30
4.7 Spectral ViT V2 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99% confidence cut 30
4.8 Spectral ViT V2 Binary Diagnostics: ROC Curve (left) and Confusion Matrix (right) 31

4.9 Max value of the output vector from the Spectral ViT V2 Binary Classification. 31

1. Introduction

When observing the night sky, astronomers frequently encounter objects that are unexpected. This occurred quite frequently in the 19th century, as astronomers armed with more powerful telescopes were probing the cosmos for the first time. In 1885 and 1895, two objects were observed in galaxies outside of our own with unique properties (de Vaucouleurs and Corwin 1985; Schaefer 1995). These objects were very bright, outshining their host galaxies, and faded away over the course of a few months. These objects were eventually named 'supernovae' (SNe) by Walter Baade and Fritz Zwicky, who were the first to suggest that these objects were the result of the death of a star, and predict the resulting object to be a cold, neutron star(Baade and Zwicky 1934). Since then, astronomers have observed thousands of SNe (Barbon et al. 1999), and have developed a classification scheme for these objects.

1.1 Supernova Classification

SNe have been traditionally classified based on their spectra starting with Minkowski (1941), who first classified SNe into two groups based on the presence or absence of hydrogen lines in their spectra. This classification provided the basis for the current

classification scheme, which is based on the presence or absence of hydrogen and silicon in the spectra of SNe. This classification scheme is shown in Figure 1.1.

Figure 1.1: Supernova classification. The classification scheme is based on the presence or absence of hydrogen and silicon features in the spectra of SNe. Figure from Turatto (2003).

SNe with hydrogen in their spectra are classified as Type II SNe, while those without hydrogen are classified as Type I SNe. Type I SNe are further subdivided into Type Ia, Ib, and Ic (Turatto 2003). Although divided into three subtypes, Type I SNe are caused by two different mechanisms. Type Ia SNe are caused by the thermonuclear explosion of a white dwarf star, while Type Ib and Ic SNe are caused by the core-collapse of a massive star (Filippenko 1997). Type Ib and Ic SNe, because of this, are actually more related to Type II SNe, and are collectively referred to as core-collapse SNe. We now know the physical origin of the spectral differences between the subtypes: stars stripped of their hydrogen envelopes are appear as Type

Ib SNe, while those stripped of both their hydrogen and helium envelopes are classified as Type Ic SNe.

1.2 Supernova Identification

The different types of SNe are caused by different mechanisms, and therefore knowing the type of a SN can provide insight into the mechanism that caused it. Merely detecting a supernova is therefore not enough; astronomers must also identify its type to identify its origin. Traditionally, a supernova is observed by several means, with discovery in photometric observations and classification using follow-up spectroscopy (e.g. Perlmutter et al. (1995)). This process is time-consuming, requiring not only the initial discovery of the supernova, but also follow-up observations to determine the type before it fades away. Today, due to the large volume of new discoveries and the small number of single and multi-object spectrographs suitable for follow-up, only about 10% of discovered SNe receive spectroscopic classification.

1.3 Dark Energy Spectroscopic Instrument

* Write a section about DESI * What is it? * Why is it important? * Why do we care about SNe here

See the intro and instrument sections of this paper: https://arxiv.org/pdf/2209.14482.pdf. You don't need to copy everything. I think 2 paragraphs should be sufficient for this section. One thing you want to emphasize is the Bright Galaxy Survey, which is where most of these transients are being observed. A question you should be able to

answer: at what redshifts are we looking for these transients? (Basically anything with $z < 0.25$.

1.4 Supernova Classification with Machine Learning

I suggest you give some background on previous approaches before jumping right into AI/ML. See the early slides in this Google slideshow.

Machine learning (ML) has been applied to many different fields for its flexibility and ability to find patterns in data. The emergence of Deep Learning (DL) has allowed ML to be applied to more and more mundane tasks, such as image classification (Krizhevsky, Sutskever, and Hinton 2012), speech recognition (Nassif et al. 2019), and natural language processing (Mikolov et al. 2013). ML and DL has also been applied to astronomy, with some success. Gauci, Zarb Adami, and Abela (2010) used a random forest algorithm to distinguish between spiral, elliptical, and irregular galaxies, and Becker et al. (2021) used a convolutional neural network (CNN) to classify the morphology of radio galaxies. CNNs found success in working within the DESI collaboration, with Parks et al. (2018) using the architecture to detect strong emission lines. These techniques have also been applied to the classification of SNe. Möller et al. (2016) used a CNN to classify SNe into Type Ia and non-Type Ia SNe.

1.5 Problem Statement

This work will explore the following questions: how can we adapt novel deep learning techniques to classify SNe in the DESI dataset? Can we use this architecture to

classify SNe into their respective subtypes accurately? And how can we decrease the amount of pre-processing required to classify these SNe accurately? In the following chapters, we describe the Deep Learning techniques adopted to answer this problem (Ch. 2); the training procedure for building our classification network (Ch. 3); and the results using simulated DESI spectra that were resampled from actual DESI measurements performed in 2021 and 2022 (Ch. 4).

2. Deep Learning Techniques

2.1 Multi-Layer Perceptrons

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 feed-forward fully connected neural networks, also known as 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 $\vec{x}^{(0)}$ with n values, an MLP applies a series of weights to each element, resulting in a linear transformation from $\vec{x}^{(0)}$ to $\vec{x}^{(1)}$, where $\vec{x}_{(1)}$ is a vector of length m. This transformation is then followed by a bias added to each element, and an activation function applied to each element. The transition from layer i to $i+1$ can be represented mathematically as

$$\vec{x}^{(i+1)} = f(\mathbf{W}\vec{x}^{(i)} + \vec{b}^{(i+1)}), \tag{2.1}$$

Figure 2.1: A simple MLP with an input dimension of 4 (green nodes), three hidden layers (blue nodes), and an output dimension of 3 (red nodes). 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).

where $\mathbf{W}$ is an $m \times n$ matrix of weights, $\vec{b}^{(i+1)}$ is a vector of length m representing the bias, and f 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, $f(x_i) = \sigma(x_i) = 1/(1+e^{-x_i})$, hyperbolic tangent, $f(x_i) = \tanh(x_i)$, and the rectified linear unit, $f(x_i) = \mathrm{ReLU}(x_i) \equiv \max(0, x_i)$, all of which are applied element-wise. The output layer of the network is a vector tailored for the particular purpose of the MLP. For example, if the network is used to classify input data $\vec{x}$ into a set of classes, the output is a vector of normalized probabilities of each class.

The MLP architecture, while simplistic, is very powerful. Cybenko (1989) showed that for any boolean function (i.e. 2-class classification), an 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. MLPs begin to show many complications when training large models, such as the exploding/vanishing gradient problem that makes them difficult to train (CITE Nielsen?), a lack of invariance under translation (important for many types of data, such as images), and exponential growth of the number of trainable parameters (Naskath, Sivakamasundari, and Begum 2022). These problems inhibit the use of MLPs on larger datasets and in more complex problems, including image recognition, time series analysis, and natural language processing. Since the mid-2000s, the weaknesses of MLPs have been addressed by the adoption of alternative network architectures such as convolutional neural networks (CNNs) and transformers.

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) were originally developed to address the lack of invariance under translation in input data (Fukushima 1979). CNNs designed for classification are traditionally composed of a series of convolutional layers, usually followed by a pooling layer, and a fully connected layer. The convolutional layers are composed of a series of filters that take a certain number of inputs from the previous layer and apply a kernel to them, resulting in a single output. These filters are applied across the entire input space, and are commonly referred to as feature maps. If N filters are applied in each convolutional layer, the result is a tensor of shape $N \times x$, where x is the shape of the original input (which can be 1D or multi-dimensional).

Figure 2.2: A convolutional filter and a 2×2 pooling applied to a 2D input. As the filter is passed over the input space, the kernel is applied. The input is padded with zeros to preserve dimensionality, which is then reduced by the max pooling. Figure adapted from Neutelings (2022).

To reduce the dimensionality of the output, a pooling layer is applied. The pooling layer takes a certain number of inputs from the previous layer, and reduces them to a single output, similar to the convolutional filters, but without the kernel application. The most common pooling layer is the max-pooling layer, which takes the maximum value in each set as the reduced value. The process of passing an input through a convolutional filter and pooling layer is shown in Figure 2.2. These convolution and pooling layers are applied in series until the desired output dimensionality is reached.

From here, depending on the task, the output of the pooling layer can be fed into a fully connected layer, similar to the MLP architecture described in Section 2.1, or it can be fed into another convolutional layer. An MLP is usually applied for classification tasks, while another convolutional layer is applied for segmentation tasks. An example of a CNN architecture used for classification is shown in Figure 2.3.

The CNN architecture was shown to be effective in image recognition tasks

Figure 2.3: A convolutional network archeticture designed for classification (adapted from Kumar 2022). Only one convolutional/pooling layer is shown, but there can be multiple depending on the application.

by LeCun, F. Huang, and Bottou (2004) and then popularized after the success of AlexNet in the ImageNet challenge Krizhevsky, Sutskever, and Hinton (2012). AlexNet was the first of many CNN architectures popularized in the 2010s for vision classification, such as VGG (Simonyan and Zisserman 2015), ResNet(He et al. 2015), and DenseNet (G. Huang et al. 2016). This also spurred a sudden increase in deep networks designed for segmentation tasks as well, such as the FCN architecture by Shelhamer, Long, and Darrell (2016). In addition to vision tasks, it also found great succes in applications such as natural language processing (Kim 2014), engineering, and medicine (Kiranyaz et al. 2021).

2.2.1 CNNs on Spectroscopic Data in Astronomy

CNNs excel at identifying patterns throughout their input space, stemming from their spacial invariance. Theoretically, this should translate to spectral classification quite well. Currently running at the National Energy Reserach Scientific Computing Center (NERSC) is a 1D CNN developed by Wasserman (2021) for DESI transient

Figure 2.4: CNN Diagnostics: ROC Curve (left) and Confusion Matrix (right)

detection. This model, while proven to be effective on training data, is limited by the number of false positivies it produces. In order to perform well on DESI data, Wasserman (2021) created a preprocessing pipeline that converts the spectra into a format that is more easily interpretable by the CNN. This preprocessing pipeline includes de-redshifting the spectra into the observed galaxies rest frame, applying a log-binning to the spectra, normalizing the flux such that the maximum flux is 1, and padding the spectra to a fixed length.

This work provides a baseline for the use of CNNs on supernovae classification which was continued by Sepeku (2022), who introduced the use of a 2D CNN for SNe classification, adding a second dimension to the spectra by re-shaping the spectra from an array of size N to an array of size $N/2 \times N/2$. The resulting 2D 'image' was then fed to a more traditional vision CNN architecture (see Appendix A.2 for a complete diagram of the architecture). This architecture was shown to train quickly (less than 1 hour on a single GPU), and was able to classify a validation set of 12888 spectra with an average precision of 0.872 via 6-way categorical classification.

Figure 2.4 shows the ROC curve and confusion matrix for the CNN trained on

synthetic spectra.

The maximum value of the output vector varied greatly between 1, which indicates a confident classification, and 0.2, which indicates a random guess. Figure 2.5 shows the distribution of the maximum value found. To ensure that the CNN is not misclassifying spectra with low confidence, a cut on the maximum value of the output vector can be applied. Figure 2.6 shows the ROC curve and confusion matrix for the CNN trained on synthetic spectra with a 99% confidence cut, resulting in 28.8% of the original validation set.

A nearly perfect ROC curve and confusion matrix are produced by the CNN when evaluated on only highly confident classifications. This indicates that the CNN is able to remedy the issue of false positives by only classifying spectra that it is confident in. However, this cut is not ideal, with only 28.8% of the data, sacrificing efficiency for purity.

The next step in training would be to either further increase the accuracy 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.

Figure 2.5: Max value of the output vector from the CNN.

Figure 2.6: CNN Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99% confidence cut

2.3 From CNNs to 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 (Figure 2.7). 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 output 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 and image classification. 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 image-appropriate alternative. In short, the input $n \times n$ image is broken into a series of $m \times m$ patches where $m < n$ and $m|n$, which are then flattened into a vector. These vectors, along with positional encodings, are then fed into the transformer architecture, as shown in Figure 2.8. For classification tasks, the first input token is replaced

Figure 2.7: Traditional transformer architecture consisting of an enocoder and decoder. Image from Vaswani et al. (2017)

with a class token. After passing through the transformer, the class token is then run through a fully connected layer to produce the final categorization probabilities. Further details are provided in Section 3.1.

2.3.1 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

Figure 2.8: Traditional ViT architecture. Purple ovals indicated positional embeddings, while pink ovals indicate the resulting tokens from the patches. Image from Dosovitskiy et al. (2020)

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 spectrum in the observers rest frame might have a continuum pattern at different locations along 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, the following series of steps were taken. First, a custom transformer architecture was created based on the traditional ViT architecture (Dosovitskiy et al. 2020), and trained on a simple synthetic dataset to validate our approach. 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 the 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 coded using the `PyTorch` deep learning framework. The architecture is shown graphically in Appendix A.3.1. The Spectral ViT is composed of three main components: the pre-processor, the encoder, and the classifier.

The pre-processing component is responsible for taking the input spectra extracted from the DESI CCDs and converting them into a series of vectors that the transformer can interpret. Consider a sample of N spectra, each with 10000 pixels corresponding to a bin containing a wavelength between 3600Å and 9800Å. Each group is split into a set number of patches (approximately 100 pixels in length). Each patch is then linearly mapped via a fully connected network to a vector 4 times smaller than the patch size. The sample is now a set of size $N \times 100 \times 25$. A classification token of the same dimension as the patches is initialized with uniformly distributed random values between 0 and 1, and then concatenated to the beginning of each sample. 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 and 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 patch index, j is the index in the hidden dimension, and the patch size is the number of pixels defined in each patch (Vaswani et al. 2017). In a CNN, the positional relationship between each pixel is understood by the convolutional layers and the pooling layers. However, without the positional embeddings, there is no difference between the first and last patch in the sample. The sinusoidal embeddings were chosen in alignment with the original transformer architecture (Vaswani et al. 2017), as well as providing a simple way to encode the positional information. A visual representation of the embeddings for the sample is shown in Fig. 3.1. Once

Figure 3.1: Visual representation of the embeddings for a sample of spectra. The x-axis represents the index of the patch, the y-axis represents the index of the hidden dimension, and the color represents the value of the embedding.

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.

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 user-defined 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 used in this work 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, defined as follows:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}. \tag{3.3}$$

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 user-defined 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 classes of supernovae.

3.1.1 Validation of Spectral ViT Architecture

After creating the Spectral ViT architecture, we tested our ability to train it effectively. A synthetic dataset, consisting of a continuum with broad Gaussian peaks placed at predetermined locations. Certain combinations of peak locations were chosen to

Figure 3.2: Synthetic spectra used to verify performance of SpectralViT. Red and green vertical lines represent features used to seperate the data into disticnt classes for SNR=2 and SNR=10, respectively. Two classes are shown for each SNR.

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. 3.2.

These datasets with SNR=10 were trivially separable by a Spectral ViT architecture with two transformer blocks, two transformer heads, eight hidden layers, and a patch size of 16 (spectra were 1024 pixels long). An accuracy of 100% on the testing set was achieved within 5 epochs of training (Figure 3.3b). Trained on the same architecture, the data with SNR=2 was more difficult to separate, only achieving a 33.975% test accuracy (Figure 3.3a).

3.2 Creation of Synthetic DESI Spectra

Once we demonstrated the Spectral ViT architecture was training effectively on simple synthetic data, the architecture was trained on DESI data. In order to create

(a) SNR = 2

(b) SNR = 10

Figure 3.3: Training of Spectral ViT on synthetic datasets. The model was able to reach 100% accuracy on the test set of SNR=10 (top), but failed to reach an accuracy of 50% on the test set of SNR=2 (bottom).

a large enough training set, observed 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 ** ** Couldn't find Eddies thesis online – ask BenZvi fora copy **

3.2.1 Pre-Processing of DESI Spectra

Once the spectra was 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

Figure 3.4: Training of the CNN on synthetic DESI spectra.

developed by ** Cite eddies paper **, and is split into 3 main steps: redshift correction, rebinning / down sampling, and normalization.

The reduced spectra, a table of best fit redshifts, and a set of fitting templates and coefficients are provided by the DESI pipeline in the form of a FITS file (Pence, W. D. et al. 2010; Guy et al. 2023). The z correction step was used to move the spectra back into the rest frame using the best-fit redshift identified 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 user-defined resolution (default 3600). The spectra were then normalized to redefine the maximum and minimum flux values to 1 and 0 respectively. Finally, for the CNN datasets, the spectra were split into a 2D array of equal height and width (i.e. for the spectra of length 3600, the array would have a shape of 60×60). After this, the spectra were split into training, testing and validation datasets that comprised 60%, 20%, and 20% of the total dataset, respectively.

3.3 Training of Neural Networks

3.3.1 CNN Training

The CNN architecture was developed by Sepeku (2022), and has properties shown in Table A.1. This CNN was trained for a maximum of 50 epochs, with a batch size of 50, and non-variable hyperparameters (Table 3.1). The timeline of the training of the CNN architecture is shown in Fig. 3.4.

3.3.2 Transformer Training

Two Spectral ViT architectures were trained on the synthetic DESI spectra. The first architecture was smaller, while the second was larger. The properties of both are shown in Table 3.2. Both were trained initially on spectra downsampled to 3600 bins. As shown in Figure 3.5, the smaller architecture was able to train effectively, reaching a test accuracy of 60.94% after approximately 31 epochs. The larger architecture was unable to train effectively, only reaching a test accuracy of around 26% before overfitting, as shown in Figure 3.6. Therefore, the smaller architecture was chosen for

Hyperparameter	Value
Learning Rate	0.00004135238172950965
Regularization	0.06
Dropout Rate	0.40904759925886294
Batch Size	50

Table 3.1: Hyperparameters of the CNN used to classify DESI spectra. CNN adapted from Sepeku (2022).

Hyperparameter	Smaller Architecture	Larger Architecture
Learning Rate	0.0001	0.01
Batch Size	16	16
Blocks	4	4
Heads	5	12
Hidden Dimension	25	72

Table 3.2: Hyperparameters of the smaller and larger Spectral ViT architecture used to classify DESI spectra.

further testing.

In addition, the smaller architecture was trained on spectra downsampled to 1800 and 900 bins to test its performance on spectra of varying resolution. As shown in Figure 3.7 and Figure 3.8, neither variation was able to train as effectively as the original binning.

The smaller architecture, with a resolution of 3600 bins, was then trained on the non redshift-corrected spectra. As shown in Figure 3.9, the model was able to train effectively, reaching a test accuracy of 49.82%. This lack of dependence on DESI redshift fitting is promising, as it means that the model can be used to classify spectra

Figure 3.5: Training of the small architecture on redshift-corrected spectra downsampled to 3600 bins. Over-fitting was determined to have occurred by Epoch 31.

Figure 3.6: Training of the large architecture on Redshift-corrected spectra downsampled to 3600 bins. No convergence above random guessing was observed.

Figure 3.7: Training of the small architecture on Redshift-corrected spectra downsampled to 1800 bins. No convergence above random guessing was observed.

Figure 3.8: Training of the small architecture on Redshift-corrected spectra downsampled to 900 bins. No convergence above random guessing was observed.

Figure 3.9: Training of the small architecture on non Redshift-corrected data downsampled to 3600 bins. Over fitting was determined to have occurred by Epoch 26.

without the need for the redshift fit.

4. Spectral ViT Model Performance

4.1 V1

The Spectral ViT's performance was evaluated on the test set of 12888 validation spectra. These spectra were not used in the training, nor the test set of the model. The resulting ROC curve and confusion matrix are shown in Figure 4.1. The ROC curve shows the model is capable of differentiating between SNe and non-SNe with an AUC of 0.99. However, differentiating between the different types of SNe is more difficult, as the confusion matrix indicates.

In contrast to the CNN previously used by Sepeku (2022), the Spectral ViT

Figure 4.1: Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right)

Figure 4.2: Max value of the output vector from the Spectral ViT V1.

V1 was more sure of its predictions, as indicated by Figure 4.2. By filtering out predictions with a maximum value less than 99%, the model's average precision was increased from 0.665 to 0.783 (Figure 4.3). This cut included 63.4% of the original predictions. Further increasing this cut to 99.9999% reduced the sample size to 5156 (40% of the original validation size), but further increased the average precision to 0.862, as shown in Figure 4.4.

While the accuracy score didn't increase significantly with the second confidence cut, the important metric is the number of positive predictions made. All AUC scores are above 0.95, and the number of found targets is significantly higher than the CNN previously used. For example, Figure 4.4 correctly classifies 4705 targets, while Figure 2.6 only correctly classifies 3742 targets under a less strict confidence

Figure 4.3: Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99% confidence cut

Figure 4.4: Spectral ViT V1 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99.9999% confidence cut

cut. This is a 26% increase in the number of correctly classified targets not excluded by the confidence cut.

4.2 V2

The goal of the Spectral ViT V2 was to see if the dependence on the redshift fit provided by the DESI pipeline could be removed. The validation set was the same size as the V1 validation set, 12888 specra. The ROC curve and confusion matrix are shown in Figure 4.5. Classification of SNe and non-SNe is still very good, with an

Figure 4.5: Spectral ViT V2 Diagnostics: ROC Curve (left) and Confusion Matrix (right)

Figure 4.6: Max value of the output vector from the Spectral ViT V2.

AUC of 1.0, but the other classes are not classified as well, resulting in an overall AUC of 0.805 and a lower average precision of 0.533.

Similar to the CNN, on a 99% confidence cut, only 34% of the original predictions remain (Figure 4.6). The AUC score and average precision both increase, as shown in Figure 4.7, but the important metric is the number of positive predictions made. The ViT, despite not looking at spectra in the rest frame, was still able to identify more Host and SN II spectra than the CNN, but not as many of the other spectra.

This, combined with the lower AUC score, indicates that the ViT V2 is not a suitable replacement for the ViT V1 for the purposes of this project. However, the ViT V2 is still able to identify SNe and non-SNe spectra with high accuracy, and could be used as a classifier for these two classes.

Figure 4.7: Spectral ViT V2 Diagnostics: ROC Curve (left) and Confusion Matrix (right) with a 99% confidence cut

Figure 4.8: Spectral ViT V2 Binary Diagnostics: ROC Curve (left) and Confusion Matrix (right)

4.2.1 Binary Classification

To test the binary classification, the probabilities of all classes apart from non-SNe were summed. This resulted in a very high AUC score of 0.997, with a nearly perfect average precision of 0.987 (Figure 4.8). This, combined with the high number of confident predictions shown in Figure 4.6, indicates the ViT V2's ability to act as a binary classifier for SNe.

Figure 4.9: Max value of the output vector from the Spectral ViT V2 Binary Classification.

Bibliography

Baade, W. and F. Zwicky (1934). "Cosmic Rays from Super-novae". In: *Proceedings of the National Academy of Sciences of the United States of America* 20.5, pp. 259–263. ISSN: 00278424. URL: http://www.jstor.org/stable/86841 (visited on 05/03/2023).

Barbon, R. et al. (Nov. 1999). "The Asiago Supernova Catalogue - 10 years after". In: 139, pp. 531–536. DOI: 10.1051/aas:1999404. arXiv: astro-ph/9908046 [astro-ph].

Becker, Burger et al. (May 2021). "CNN architecture comparison for radio galaxy classification". In: 503.2, pp. 1828–1846. DOI: 10.1093/mnras/stab325. arXiv: 2102.03780 [astro-ph.GA].

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.

de Vaucouleurs, G. and Jr. Corwin H. G. (Aug. 1985). "S Andromedae 1885: A Centennial Review". In: 295, p. 287. DOI: 10.1086/163374.

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.

Filippenko, Alexei V. (Jan. 1997). "Optical Spectra of Supernovae". In: 35, pp. 309–355. DOI: 10.1146/annurev.astro.35.1.309.

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.

Gauci, Adam, Kristian Zarb Adami, and John Abela (May 2010). "Machine Learning for Galaxy Morphology Classification". In: *arXiv e-prints*, arXiv:1005.0390, arXiv:1005.0390. DOI: 10.48550/arXiv.1005.0390. arXiv: 1005.0390 [astro-ph.GA].

Guy, J. et al. (Mar. 2023). "The Spectroscopic Data Processing Pipeline for the Dark Energy Spectroscopic Instrument". In: *The Astronomical Journal* 165.4, p. 144. DOI: 10.3847/1538-3881/acb212. URL: https://doi.org/10.3847/1538-3881/acb212.

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

Huang, Gao et al. (2016). *Densely Connected Convolutional Networks*. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993.

Kim, Yoon (2014). *Convolutional Neural Networks for Sentence Classification*. DOI: 10.48550/ARXIV.1408.5882. URL: https://arxiv.org/abs/1408.5882.

Kiranyaz, Serkan et al. (2021). "1D convolutional neural networks and applications: A survey". In: *Mechanical Systems and Signal Processing* 151, p. 107398. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398. URL: https://www.sciencedirect.com/science/article/pii/S0888327020307846.

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.

Kumar, Ajitesh (Apr. 2022). *Different types of CNN Architectures explained: Examples*. URL: https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/.

LeCun, Y., F. 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.

Mikolov, Tomas et al. (Jan. 2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv e-prints*, arXiv:1301.3781, arXiv:1301.3781. DOI: 10.48550/arXiv.1301.3781. arXiv: 1301.3781 [cs.CL].

Minkowski, R. (Aug. 1941). "Spectra of Supernovae". In: 53.314, p. 224. DOI: 10.1086/125315.

Möller, A. et al. (Dec. 2016). "Photometric classification of type Ia supernovae in the SuperNova Legacy Survey with supervised learning". In: *Journal of Cosmology and Astroparticle Physics* 2016.12, pp. 008–008. DOI: 10.1088/1475-7516/2016/12/008. URL: https://doi.org/10.1088/1475-7516/2016/12/008.

Naskath, J., G. Sivakamasundari, and A. Alif Siddiqua Begum (Oct. 2022). "A Study on Different Deep Learning Algorithms Used in Deep Neural Nets: MLP SOM and DBN". In: *Wireless Personal Communications* 128.4, pp. 2913–2936. DOI:

10.1007/s11277-022-10079-4. URL: https://doi.org/10.1007/s11277-022-10079-4.

Nassif, Ali Bou et al. (2019). "Speech Recognition Using Deep Neural Networks: A Systematic Review". In: *IEEE Access* 7, pp. 19143–19165. DOI: 10.1109/ACCESS.2019.2896880.

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

— (Apr. 9, 2022). *Neural networks*. URL: https://tikz.net/conv2d/.

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.

Pence, W. D. et al. (Nov. 2010). "Definition of the Flexible Image Transport System (FITS), version 3.0". In: *A&A* 524, A42. DOI: 10.1051/0004-6361/201015362. URL: https://doi.org/10.1051/0004-6361/201015362.

Perlmutter, S. et al. (Feb. 1995). "A Supernova at Z = 0.458 and Implications for Measuring the Cosmological Deceleration". In: 440, p. L41. DOI: 10.1086/187756. arXiv: astro-ph/9505023 [astro-ph].

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

Schaefer, Bradley E. (July 1995). "The Peak Brightness of SN 1895B in NGC 5253 and the Hubble Constant". In: 447, p. L13. DOI: 10.1086/309549.

Sepeku, Edmund (May 2022). "–". Bachelor's Thesis. Rochester, NY: University of Rochester.

Shelhamer, Evan, Jonathan Long, and Trevor Darrell (2016). *Fully Convolutional Networks for Semantic Segmentation*. DOI: 10.48550/ARXIV.1605.06211. URL: https://arxiv.org/abs/1605.06211.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*.

Turatto, M. (2003). "Classification of Supernovae". In: *Supernovae and Gamma-Ray Bursters*. Ed. by K. Weiler. Vol. 598, pp. 21–36. DOI: 10.1007/3-540-45863-8_3.

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.

Wasserman, Amanda (May 2021). "Using Machine Learning to Identify Transients in the DESI Survey". Bachelor's Thesis. Rochester, NY: University of Rochester.

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

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_spec1 (InputLayer)	[(None, 60, 60, 1)]	0	
conv2d_1 (Conv2D)	(None, 53, 53, 5)	325	input_spec1[0][0]
conv2d_3 (Conv2D)	(None, 29, 29, 5)	5125	input_spec1[0][0]
conv2d_4 (Conv2D)	(None, 5, 5, 5)	15685	input_spec1[0][0]
conv2d (Conv2D)	(None, 57, 57, 5)	85	input_spec1[0][0]
batch_normalization_1 (BatchNor	(None, 53, 53, 5)	212	conv2d_1[0][0]
batch_normalization_3 (BatchNor	(None, 29, 29, 5)	116	conv2d_3[0][0]
batch_normalization_4 (BatchNor	(None, 5, 5, 5)	20	conv2d_4[0][0]
batch_normalization (BatchNorma	(None, 57, 57, 5)	228	conv2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 17, 17, 5)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 45, 45, 5)	1285	input_spec1[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 9, 9, 5)	0	batch_normalization_3[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 1, 1, 5)	0	batch_normalization_4[0][0]
flatten (Flatten)	(None, 16245)	0	batch_normalization[0][0]
flatten_1 (Flatten)	(None, 1445)	0	max_pooling2d_1[0][0]
flatten_2 (Flatten)	(None, 10125)	0	conv2d_2[0][0]
flatten_3 (Flatten)	(None, 405)	0	max_pooling2d_3[0][0]
flatten_4 (Flatten)	(None, 5)	0	max_pooling2d_4[0][0]
concatenate (Concatenate)	(None, 28225)	0	flatten[0][0] flatten_1[0][0] flatten_2[0][0] flatten_3[0][0] flatten_4[0][0]
dense (Dense)	(None, 64)	1806464	concatenate[0][0]
dropout (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 6)	390	dropout[0][0]

Total params: 1,829,935
Trainable params: 1,829,647
Non-trainable params: 288

Table A.1: Architecture of the CNN used to classify DESI spectra. CNN adapted from Sepeku (2022).

B. Title of Appendix B