Soft-Classification of Galaxy Morphologies Using Deep Neural Networks

Candidate 33770, Candidate 30855, Candidate 26527, Candidate 26684

Abstract

This report proposes several contributions to the existing literature on galaxy image classification. We introduce constraints into our loss function that help our models better emulate responses when classifying galaxy morphologies by image. We also introduce a modified Transformer-based architecture (Vision Transformer) can perform competitively versus convolutional neural networks (CNNs) in such tasks.

1. Introduction

Galaxies exhibit diverse features and structures, including irregular, elliptical, and spiral morphologies. Astrophysicists leverage these morphological structures to investigate theories on galaxy evolution and formation (Buta, 2011; Tinsley, 2022). However, such research first requires galaxies to be classified by their morphology. Traditionally, this involved manual observation of telescope-captured images. Over the past decade, the number of these images has exploded, leading researchers to resort to crowd-sourced initiatives such as the Galaxy Zoo Project (Willett et al., 2013) to classify the images using human volunteers. Yet, with the exponential growth in image volume, manual classification is no longer viable (Guruprasad, 2023). We aim to utilise deep learning architectures to automatically analyse JPEG images of galaxies and determine the probability that each galaxy has a certain morphology. We aim for our models to classify galaxies morphologies as well as humans, thus allowing galaxy image classification to be scaled up significantly while saving human labour.

We assessed the performance of four different deep learning architectures applied to the task of galaxy morphological classification. The first is a custom-designed CNN, the second is a Transformer-based architecture, and the other two are fine-tuned CNNs based on well-known architectures (VGG16 and ResNet50). The choice of our model architecture was informed by our view that a good model needs to incorporate the broad or global features of an image, as a galaxy's morphology is often represented by such features. We introduced a custom loss function to our model training, which uses constraints that help our models better emulate human responses. In out-of-sample testing, the custom CNN performed the best followed by our Transformer-based model. These two models would have placed in top 12% of the Kaggle Galaxy Zoo Challenge - a well known galaxy morphology classification competition, earning them a silver medal. In contrast, the fine-tuned CNNs performed significantly worse.

2. Classification Probabilities From the Galaxy Zoo Decision Tree

The Galaxy Zoo Project guides its volunteers through the classification process of a galaxy based on questions structured through a decision tree. The decision tree consists of 11 questions, with each question having 2-7 responses. This leads to a total of 37 responses.

Each galaxy's classification is the result of a specific path down the decision tree. Multiple volunteers (typically 40-50) all classified the same galaxy, resulting in multiple paths, and these paths generate probabilities for each node. Volunteers begin with general questions (e.g. *is it smooth?*) and move on to more specific ones (e.g. *how many spiral arms are there?*). A graphical illustration of the decision tree is shown in Figure 1. A full list of the questions are provided in Appendix A.1.

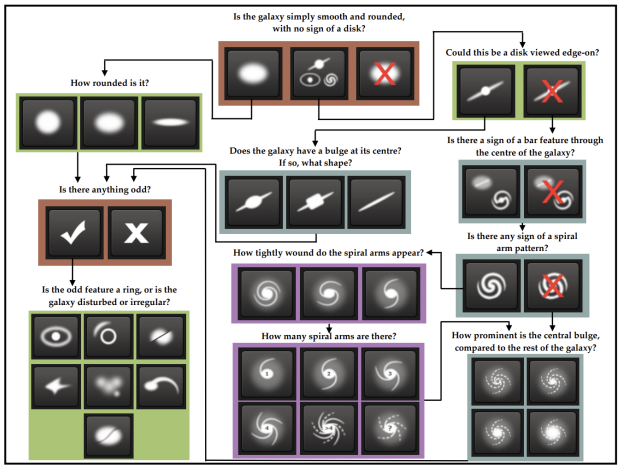

Figure 1. Galaxy Zoo Decision Tree (Willett et al., 2013)

At each node or question, the total initial probability of a classification will sum to 1. Those initial probabilities are then weighted based on previous questions. For the first set of responses in Question 1 (smooth, features/disk, star/artifact), the values in each category are simply the likelihood of the galaxy falling in each category. These values sum to 1. For each subsequent question, the probabilities are first computed (these will sum to 1) and then multiplied by the value which led to that new set of responses.

For example, suppose a galaxy had 80% of users identify it as smooth, 15% as having features/disk, and 5% as a star/artifact. The probabilities for the first three responses will be $q_{1.1} = 0.8$, $q_{1.2} = 0.15$, and $q_{1.3} = 0.05$.

Based on the decision tree, for the 80% of volunteers that identified the galaxy as "smooth", they will be brought next to Question 7 to record responses for the galaxy's relative roundness. Suppose these votes were for 50% completely round, 25% in-between, and 25% cigar-shaped. The output values corresponding to the responses to Question 7 are thus:

$$q_{7.1} = q_{1.1} \times 0.50 = 0.40$$
$$q_{7.2} = q_{1.1} \times 0.25 = 0.20$$
$$q_{7.3} = q_{1.1} \times 0.25 = 0.20$$

This method of cumulatively multiplying probabilities applies for every morphology class, as mapped by Figure 1. The sum of response probabilities in Question 1.1 - 1.3 for each galaxy will always sum to 1, since this question is answered for every galaxy. Responses for Questions 6.1 and 6.2 have also been normalized to sum to 1. For the remaining questions, the sum of responses is ≤ 1. The 37 probabilities for each response will then be concatenated into a 37×1 target vector for each galaxy. The objective of our models is to generate 37×1 vectors that match the human-derived probabilities as closely as possible, with galaxy JPG images as the input.

Due to the cumulative multiplying of probabilities and the questions being structured into a decision tree, the response values are not independent of one another. In Section 6.2, we discuss how we exploited these dependencies among the response values to improve our model training.

3. Literature Review

3.1. Kaggle Galaxy Zoo Challenge

In 2014, the data science competition platform Kaggle held the Galaxy Zoo Challenge (AstroDave, 2013), where participants competed to build the best model to match human response probabilities when classifying galaxy morphologies by their images. Most top-ranked solutions utilised CNN-based models, which were at the time state-of-the-art in image recognition tasks. The winning solution came from DeepMind researcher Sander Dieleman (Dieleman, 2014). At a high level, his CNN architecture consisted of 4 convolutional layers and 3 dense layers. All convolutional layers included a ReLU activation. The first, second and fourth convolutional layers are followed by 2×2 max-pooling. The convolutional filter sizes decrease gradually with each layer, starting out as 6×6, before shrinking to 5×5, and then 3×3. The dense layers consists of two layers with 2048 units each, and a final output layer with 37 units. The CNN architecture had 42 million parameters, and took 67 hours to train. It achieved a root mean squared error (RMSE) of 0.0749, the lowest-recorded error in the competition. Beyond that, (Dieleman, 2014) also incorporated two noteworthy methods in his solution:

1. **Parallel image augmentation**: Galaxy images are passed through an image augmentation routine which increases parameter sharing in the neural network, leading to better generalisation. Each galaxy image is divided into 16 smaller segments through a series of cropping, downsampling, rotating, and flipping operations, exploiting the rotation invariance of the images (More details in Section 4.4). Each segment was processed using the same CNN architecture, resulting in a 16x increase in parameter sharing, reducing overfitting. The information from all 16 segments is combined by concatenating the outputs and feeding them to subsequent dense layers. This technique improves the model's ability to analyze key features concentrated at galaxy centers, aiding in galaxy identification.

2. **Model ensembling**: The base CNN architecture was used to create multiple variants with different filter size combinations, different regularisation techniques, and different dense layer configurations. The output from this ensemble of model variants were then averaged over to yield a final output. Ensembling is a very well-known technique in the machine learning space, where outputs from multiple models are combined together, with the aim of improving predictive performance (Freund & Schapire, 1997; Breiman, 2001).

We leveraged some ideas from (Dieleman, 2014). For example, we employed similar image augmentation techniques in our model training. The idea of segmenting images into smaller segments and learning from them also partially inspired our choice to use a Transformer-based architecture - which does something similar - as an alternative to CNNs for galaxy image classification (more details in Section 5.3). We were less able to leverage other ideas like the actual CNN architecture and model ensembling due to computational constraints.

3.2. Other Related Work

Over the years since the Galaxy Zoo Challenge, CNN architectures have gotten much more advanced. This has led researchers to incorporate newer architectures to galaxy image classification tasks. A non-exhaustive list of some approaches are given below:

1. (Walmsley et al., 2020) utilised Bayesian CNNs based on Monte Carlo dropouts and a novel generative model of Galaxy Zoo volunteer responses to infer posterior probabilities for galaxy morphologies based on their images.

2. (Dai & Tong, 2019) applied a variation of the ResNet architecture (He et al., 2015) to the Galaxy Zoo data. Unlike Kaggle's Galaxy Zoo Challenge, their model was not trained to output a vector that best matches human response probabilities. Rather, the model performs 5-way classification of galaxy images (completely round smooth, in-between smooth, cigar-shaped smooth, edge-on, spiral).

3. (Lin et al., 2022) used Transformer-based architectures to perform 8-way classification of galaxy images (round elliptical, in-between elliptical, cigar-shaped elliptical, edge-on, barred spiral, unbarred spiral, irregular, merging) using Galaxy Zoo data.

4. (Schneider et al., 2023) assessed the effectiveness of fine-tuning a pre-trained AlexNet CNN (Krizhevsky et al., 2012) to perform 4-way classification of galaxy images (elliptical, lenticular, spiral, irregular) using Galaxy Zoo data.

3.3. Our Contributions

Below, we list two contributions from our report, which to the best of our knowledge, are novel to the existing literature on galaxy image classification.

1. **Constrained loss function**: We exploited the dependencies among the Galaxy Zoo response probabilities to derive a set of constraints on the model output, which we used to formulate a custom loss function for model training. While (Dieleman, 2014) did impose constraints on his model outputs as well, those constraints are not explained in detail, and we believe that our constraints follow more naturally from the decision tree structure of the Galaxy Zoo questions. More details in Section 6.2.

2. **Transformer-based architecture**: We explored the use of Transformer-based architectures as an alternative to CNNs for classifying galaxy morphologies. While previous work has used Transformer-based architectures for galaxy image classification (Lin et al., 2022), our approach is, to the best of our knowledge, the first time anyone has used a Transformer-based architecture to attempt matching human response probabilities, as in the Galaxy Zoo Challenge format. More details in Section 5.3.

4. Data

4.1. Description

The dataset was taken from Kaggle's Galaxy Zoo competition site. It comprised 61,578 coloured images in JPEG format - each with a resolution of 424×424 pixels - plus a table of target vectors. Each galaxy image has an associated 37×1 target vector corresponding to the output values from the 37 responses. All entries in the 37×1 target vector are floating point numbers between 0 and 1. These represent the probabilities assigned to the morphology of the galaxy based on the questions from the decision tree, with the probabilities coming from crowdsourced volunteer classifications as part of the Galaxy Zoo project. A high number (close to 1) indicates that a large proportion of users identified this morphology in the image. Low numbers for a category (close to 0) indicate the feature is likely not present. By considering the target vector as a whole, we get an idea of the various morphological structures present in the galaxy image.

4.2. Test-Training Split

We employed a train-validation-test split modelling. 64% of images were assigned to the training set, which underwent data augmentation as described in Section 4.4. 16% were used as a validation set and did not undergo augmentation. By not augmenting the validation set, we ensure that the performance evaluation reflects the model's true ability to identify morphologies on real image variations; the augmented training set presents a more challenging environment for the model to learn from. The remaining 20% is kept as test data, to assess the models' out-of-sample performance in Section 7.

4.3. Data Pre-processing

Represented as a NumPy array, the images required approximately 33GB of RAM (A.4 in appendix), rendering it computationally impractical to load into memory in its entirety. To address this challenge, we applied a series of data reduction pre-processing steps, ultimately reducing the in-memory size to 757MB. The pre-processing steps included:

1. Center-cropping the images to a resolution of 210×210 pixels.

2. Resampling the images to a resolution of 64 × 64 pixels using bicubic interpolation (Xia et al., 2013).

Center-cropping was used since the galaxies, the subjects of interest, are centrally located in the images. The surrounding void space does not contribute to the identification of galaxies, thus its removal is beneficial. The downsampling process further reduced the input size, making it more manageable for processing. Additionally, we explored data streaming solutions using TensorFlow. However, we encountered I/O limitations when streaming from Google Drive to Colab. We retained the list of filenames, which include the galaxy IDs corresponding to the target vectors, and used this information to assemble the target set in matching order. Figure 2 shows a sample of images before and after pre-processing.

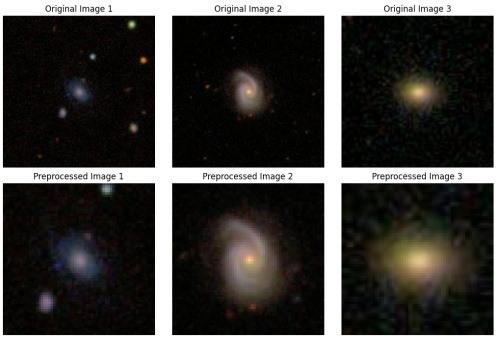

Figure 2. Galaxy images pre-versus-post-processing

4.4. Data Augmentation

As briefly mentioned in Section 3, galaxy images exhibit rotation invariance. This means that irrespective of how we rotate an image, the galaxy's morphological properties do not change. Furthermore, if we apply affine transformations (excluding skew) to the image while ensuring that the galaxy remains within the image boundaries, the galaxy's morphology doesn't change (Fisher et al.). We exploited these properties to mitigate overfitting and improve model performance with a richer training set. Each time an image was passed to the network as input, a combination of the following random transformations was applied, such that the model never saw the same image twice. Any newly introduced pixels were filled using a nearest-neighbours approach.

1. **Zoom**: Randomly zoom in or out of the image in the range $\pm 10\%$.

2. **Rotation**: Randomly rotate the image by angles in the range $\pm 25°$.

3. **Width Shift**: Shift the image horizontally by up to $\pm 5\%$ of its width.

4. **Height Shift**: Shift the image vertically by up to $\pm 5\%$ of its height.

5. Model Architectures

5.1. Model Design Intuition

Our model architecture choices were guided by the unique characteristics of galaxy images, which are defined by broader features compared to standard images. For instance, while classifying a cat photo relies more on localized features (ears, tail, nose), galaxy morphologies (round, irregular, spiral) are more broader features. This need to emphasise broader features was what led us to opt for the parallel CNN and Transformer-based model architectures for galaxy image classification, as we describe in further detail below.

5.2. Parallel CNN

We designed a custom CNN architecture that uses parallel convolution pathways to extract both broad and fine spatial features in the galaxy images. By employing both large (6×6) and small (3×3) convolution filters in separate dual paths, we hope to capture both broader and localised features in galaxy images relevant to its morphologies, as per our intuition in Section 5.1. More detail on our model design decisions is provided below. Figure 3 contains a schematic diagram of the model architecture.

The first convolutional path employs 6×6 filters, aimed at capturing wide spatial patterns in the images. Each convolution layer utilises the ReLU (Rectified Linear Unit) activation function, known for its efficiency in facilitating non-linear learning (Agarap, 2019). This path incorporates batch normalisation to standardise the inputs of each layer, thereby stabilising the learning process. Following this, max pooling is utilised to reduce the spatial dimensions, improving computational efficiency. A dropout layer is then applied.

The second convolutional path utilises 3×3 filters, focusing on the extraction of finer, more subtle galactic features. The design of this path is the same, ensuring an identical approach towards feature extraction, dimensionality reduction, and regularisation. Padding was used for both paths such that we align the dimensions for further convolutions in the next stage.

After the feature extraction paths, the outputs from the parallel pathways are concatenated, creating a comprehensive feature set. This combined feature set is further processed through 3 more successive convolution layers. Each convolution layer uses a filter of size 3×3 with ReLU activation, followed by a max pooling, and then a dropout layer.

The network concludes with a sequence of dense layers with ReLU activation that also implement L2 regularisation. Another dropout layer is employed before final output layer. The final output layer is of size 37 and uses sigmoid activation, which output a 37-dimensional vector of probabilities for the question responses. Note that unlike typical image classification CNNs that use a softmax activation on the final layer, we use a sigmoid function as the probabilities in the output vector do not sum to 1, while each response probability is bounded between 0 and 1. Our model consists of 1.2 million trainable parameters.

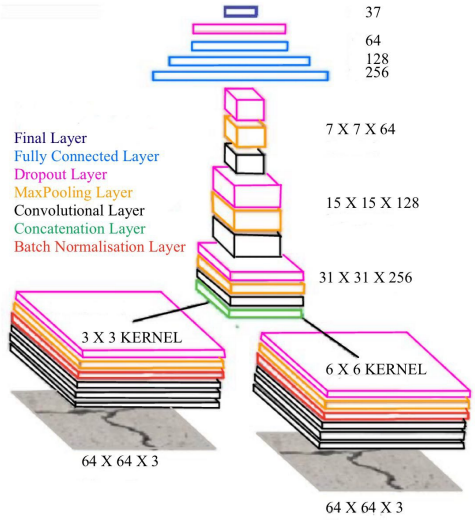

Figure 3. Parallel CNN architecture

5.3. Vision Transformer

The Vision Transformer (ViT) is a Transformer-based architecture designed specifically for image recognition tasks, developed by Google Research staff (Dosovitskiy et al., 2021). At a high level, the ViT works by treating images as sentences. It segments images into sequential image patches (similar to how sentences are broken down into word tokens). These image patches are then projected into lower dimensional embeddings and passed through a multiple transformer encoders, similar to how word tokens are processed. We thought that the ViT's self-attention mechanism would allow it to better learn the global features in the galaxy images, as it can potentially better account for the relationship between distant image patches, as per our intuition in Section 5.1. The ViT's concept of learning from smaller parts of the image also bears some similarity to (Dieleman, 2014)'s approach to the Kaggle Galaxy Zoo competition, which was another reason we chose this architecture.

In (Dosovitskiy et al., 2021), the base ViT model's transformer encoder was very similar to the original transformer architecture by (Vaswani et al., 2023). The base model consisted of 12 transformer encoder blocks, each consisting of the following components.

1. An initial layer norm.
2. A multi-headed self-attention layer with 8 heads.
3. A second layer norm.
4. A 2-layer (multi-layer perceptron) MLP of size 3072 with GeLU activation (Hendrycks & Gimpel, 2023), with output layer of size D.
5. Skip connections linking the input to the head of the multi-headed self-attention layer, and the MLP layer.

The model was trained on 224×224 images. Image recognition is done through the following procedure:

1. Segment a 224×224 image into multiple patches of size 16×16.
2. Flatten the patches and project them into lower dimensional embeddings of $D = 768$.
3. Incorporate learnable positional embeddings to take account of the positions of the patches within the image.
4. Attach a learnable classification token of size D to the head of the sequence of embeddings. The attachment of a learnable classification token was similar to the procedure done by the BERT model (Devlin et al., 2019).
5. Pass sequence of embeddings into the 12 layers of transformer encoder blocks. Extract the classification token from the output of the final transformer layer and pass it through a final MLP consisting of a hidden layer of size D with GeLU activation, plus an output layer whose size equals to the number of image classes, with softmax activation.

The base ViT had 86 million parameters. A graphical illustration of the ViT architecture and the image recognition procedure is provided in Figure 4.

Initially, we hoped to perform transfer learning on a pretrained ViT, but early-stage training showed that its performance was subpar. As such, we decided to build and train our ViT from scratch, based on the source code by (Salama, 2021). We made several modifications to our ViT architecture so that our model has a more manageable 1.8 million trainable parameters. The modifications are listed below.

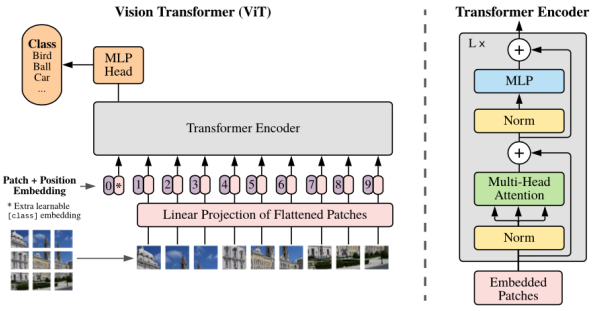

Figure 4. ViT architecture (Dosovitskiy et al., 2021)

1. Instead of 224×224 images, our images were of size 64×64, split into patches of size 8×8.

2. Our patches were projected into embeddings of smaller dimension, with $D = 64$.

3. We reduced the size of the MLP in our transformer encoders, with our MLPs having 2 layers of size 128 and 64 respectively.

4. We reduced the size of our final MLP at the head, with the MLP consisting of two hidden layers of size 64, and a final output layer of size 37. Instead of a softmax activation function in the final output layer, ours had a sigmoid activation function, for the same reason as explained in Section 5.2.

5.4. Pre-Trained CNNs

We fine-tuned two well known pre-trained CNNs, which act as a performance baseline to compare our parallel CNN and ViT:

1. **VGG16**: Developed by the Visual Geometry Group at the University of Oxford (Simonyan & Zisserman, 2015), VGG16 comprises 21 layers (13 convolutional layers, 3 dense layers, 5 max-pooling layers). It features a repetitive pattern of 3×3 convolution layers followed by max-pooling layers stacked sequentially, with the dense layers forming a multi-layer perceptron (MLP) at the head. For fine-tuning, we first imported a base VGG16 model pre-trained on the ImageNet dataset (Deng et al., 2009). We then froze the convolutional layers in the base model, and replaced the original MLP at the head with a trainable 2-layer MLP. Our MLP consists of a size 512 hidden layer with ReLU activation, and an output layer of size 37 with sigmoid activation. Our modified VGG16 architecture had 27.5 million parameters in total, out of which 12.9 million are trainable.

2. **ResNet50**: Developed by researchers at Microsoft Research (He et al., 2015), ResNet50 is comprised of multiple convolutional blocks with 3×3 filters and an MLP at the head. It also contains skip connections between the convolutional layers to mitigate the problem of exploding/vanishing gradients. For fine-tuning, we first imported a base ResNet50 model pre-trained on the ImageNet dataset. We then froze the convolutional layers in the base model, and replaced the original MLP at the head with a trainable 2-layer MLP. Unlike VGG16 above, the output of the final convolutional layer was too big if it was flattened and immediately passed into a dense layer. So we inserted a 2×2 global max-pooling layer before the MLP to reduce the dimentionallity of the final convolutional layer's output. This enabled us to increase the size of our hidden MLP layer to 2048 (with ReLU activation), followed by an output layer of size 37 with sigmoid activation. Our modified ResNet50 architecture had 27.9 million parameters in total, out of which 4.3 million are trainable.

6. Model Training

6.1. Evaluation

As in the Galaxy Zoo Challenge, we use Root Mean Squared Error (RMSE) as the evaluation metric:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(p_n - a_n)^2} \tag{1}$$

We aim to minimise the RMSE, which is calculated based on the predicted response probability (p_n) and actual response probability (a_n). N equals the total number of galaxies multiplied by the 37 responses. Intuitively, what we want is for the model to produce an output vector that matches human response probabilities as closely as possible for a given galaxy image.

6.2. Custom Loss Function With Constraints

As explained in Section 2, the response values in the target vectors are not independent of one another. For instance, we know that the first 3 probabilities of the target vector must sum to 1, as they correspond to the responses in Question 1 (*Is the object a smooth galaxy, a galaxy with features/disk or a star?*) which lie on top of the decision tree. As per the example given in Section 2, we know that probability values corresponding to the responses in Question 7 (*How round is the smooth galaxy?*) must sum up to equal the Question 1.1 probability value (i.e. the percentage of people classifying the galaxy as "smooth"). Using such knowledge, we can build specific constraints derived from these interdependencies and incorporate them into our model training. In total, we identified 11 constraints (See Appendix A.2).

We created a custom loss function that is equal to the sum of the mean squared error (MSE) and absolute deviations from all 11 constraints, as per Equation 2. Categorical cross-entropy is not suitable here, since it requires the output probabilities to sum to 1. MSE is free of this constraint, hence why we included it in our loss function.

$$L = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{37}\sum_{j=1}^{37}(p_{ij} - \hat{p}_{ij})^2 + \lambda\sum_{k=1}^{11}|c_{ik}|) \quad (2)$$

Where N is the number of images in the current minibatch, p_{ij} is the probability in the j^{th} position of output vector i, and $|c_{ik}|$ is the absolute deviation of output vector i from the k^{th} constraint. Our models were then trained to minimise the custom loss function, allowing the incorporation of the Galaxy Zoo decision tree structure into the model training itself. λ is a constant that determines how we should weight the losses from constraint deviations relative to the MSE. We experimented with different values of λ, and found that $\lambda = 0.001$ yielded the best training performance. See Figure 7 in the Appendix for a demonstration of this result.

6.3. Parallel CNN Training

The parallel CNN was trained using the training and validation data as described in Section 4.2. We also employed the custom loss function from Section 6.2, training the parameters using an Adam optimiser (Kingma & Ba, 2017) with default hyperparameters (learning rate $= 0.001$, $\beta_1 = 0.09$, and $\beta_2 = 0.999$). Training was conducted on a single Nvidia V100 GPU over 300 epochs. An early stopping callback was used, programmed to halt training after 50 epochs without improvement. Consequently, the training concluded after 182 epochs, with the optimal weights being those from epoch 132. Dropout layers of 0.1 for convolution layers and 0.2 for dense layers were incorporated. Furthermore, L2 regularisation ($\lambda = 0.01$) was applied to dense layers. In Figure 8, we see that the model is slow to overfit due to these measures. The training and validation RMSE of the best model were 0.0969 and 0.0976 respectively.

6.4. ViT Training

Our ViT was trained using the same training and validation data described above. Similar to the parallel CNN, we trained our ViT using our custom loss function, over 300 epochs using a single Nvidia V100 GPU. Training was done using an Adam optimiser, with a default learning rate value of 0.001, $\beta_1 = 0.09$, and $\beta_2 = 0.999$. The choice of β_1 and β_2 was the same configuration used in ViT's original paper (Dosovitskiy et al., 2021). The authors did use a learning rate with a linear warm up and decay. However, the time steps for the warm up and decay was determined by their training dataset size, which was less applicable to us as we had a different training dataset. Similar to the original paper, dropouts of 0.1 are applied after every dense layer except for the ones used for embeddings. In smaller experiments, we noticed that the biggest marginal improvements to the validation RMSE came from increasing the training epochs, rather than tweaking the optimiser and regularisation parameters. As such, we decided to focus our limited computational resources on stretching out the number of training epochs, rather than using up too much computational resource on hyperparameter tuning. An early stopping callback of 50 epochs was used as well. The training concluded without early stopping, with little evidence of overfitting as seen by Figure 9. The training and validation RMSE of the best model were 0.1045 and 0.0983 respectively.

6.5. Fine-tuning Pre-Trained CNNs

As the imported base models for VGG16 and ResNet50 were pre-trained on ImageNet images that are of size 224×224, we deemed that it would be sub-optimal to pass our 64×64 galaxy images directly into the models for training. Therefore, we wrote a function to resize our galaxy images back to 224×224 in batches, before passing them into the models. Given the extra memory taken up by larger images, and the size of the pre-trained CNNs (VGG16 and ResNet50), it took roughly 5 times longer to train each epoch for both models compared to our parallel CNN and ViT. As such, we opted to train on a smaller number of epochs - 50 - using a single Nvidia V100 GPU. Training was done using an Adam optimiser with default hyperparameters (learning rate value of 0.001, $\beta_1 = 0.09$, and $\beta_2 = 0.999$). In smaller scale experiments, we found that different optimisers (SGD, RMSProp) or learning rate hyperparameters did not add much value, hence the decision to stick to the Adam optimiser with default hyperparameters. Dropouts of probability 0.1 are applied to all trainable layers on both models. An early stopping callback of 20 epochs was used as well. For both models, training concluded without early stopping, as shown in Figure 10 and Figure 11. For the fine-tuned VGG16, the training and validation RMSE of the best model was 0.1171 and 0.1134 respectively. For the fine-tuned ResNet50, it was 0.1473 and 0.1403.

7. Results

Our best model was the parallel CNN, with an RMSE of 0.0985 on the test set, followed by the Vision Transformer with an RMSE of 0.0995. These scores would place us in the top 12%, with a silver medal classification in the Galaxy Zoo Challenge. The fine-tuned VGG16 and ResNet50 performed significantly worse, with RMSEs of 0.1152 and 0.1433 respectively. Interestingly, the ViT had an RMSE that was competitive with the parallel CNN, highlighting the potential of Transformer-based architectures in image classifica-

tion tasks. Another interesting observation was the subpar performance of the pre-trained VGG16 and ResNet50 models, given the size of their parameter set and original training data. We think there are two reasons they underperformed. The first was that they were trained on a lower number of epochs (50 epochs versus 300 for parallel CNN/ViT), as training time per epoch was considerably longer. The second was that the pre-trained weights in their convolutional layers were not well-trained to capture broader features in the galaxy images. While far from being conclusive, the results supported our hypothesis in Section 5.1 that broader features do matter proportionally more in classifying galaxy morphologies. The results also suggest that longer training length was a more decisive factor in improving performance relative to increased model complexity, as the parallel CNN and ViT was much smaller compared to VGG16 and ResNet50.

Model Architecture	Out-of-Sample RMSE
Parallel CNN	**0.0985**
ViT	0.0995
VGG16	0.1152
ResNet50	0.1433

Table 1. RMSE scores on the test set

8. Model Explainability Using Grad-CAM

Since our best performing model was the parallel CNN, we focused on better understanding whether the model was interpreting the galaxy images in an intuitive manner. This is assessed using Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2017). We first divided our model's 37×1 output vector into 11 distinct segments. Each segment contains the response probabilities for each of the 11 questions in the Galaxy Zoo decision tree. We then took the highest probability response and established its relationship with the gradients in the final convolution layer.

Figure 5 provides an example. It contains the image of a disk-shaped galaxy. When we inspect the Grad-CAM heatmap associated with the first question (*"Is the galaxy simply smooth/rounded, or a disk, or a star/artefact?"*), we see that the model is focussing on the top half of the galaxy to understand its shape. Furthermore, when we assess the Grad-CAM heatmap associated with the follow-up question (*"Could this be a disk viewed edge on"*), Figure 6 shows the model focusing on the edge of the galaxy, which is what we would intuitively expect. This behaviour suggests that the model can infer the morphologies of galaxies based on particular features in the galaxy images, depending on what question is being asked.

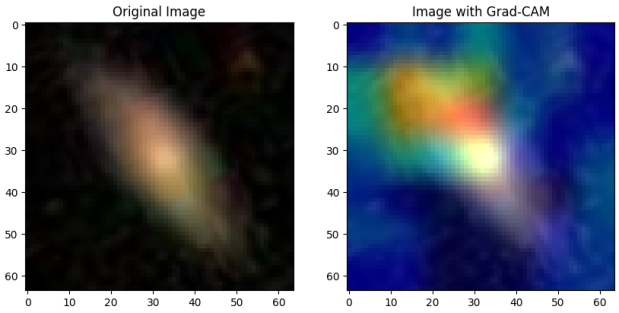

Figure 5. Grad-CAM (Is the galaxy simply smooth/rounded, or a disk, or a star/artifact?)

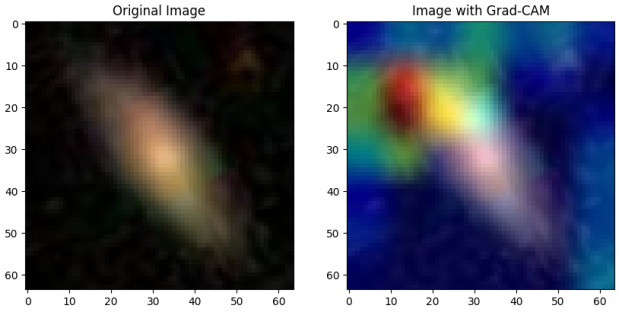

Figure 6. Grad-CAM (Could this be a disk viewed edged on?)

9. Conclusion

Our report introduces several novel contributions to the existing literature concerning galaxy morphological classification, namely the use of constraints that exploit dependencies among human response probabilities, and the application of Transformer-based architectures to the task. We also show that Transformer-based architectures can perform just as well as CNNs, and that smaller custom-designed architectures can outperform much larger fine-tuned CNNs in galaxy morphology classification.

Further Work: One area of further study is to see whether fine-tuning pre-trained CNNs over a larger number of training epochs would improve performance. It may also be worth examining whether fine-tuning newer Vision Transformer models such as ConvNeXt and Mobile-Former (Chen et al., 2022; Liu et al., 2022) would lead to better performance. Another avenue for research is to conduct wider hyperparameter tuning studies of the parallel CNN and ViT models, to see whether performance could be improved further. A final area of research is to examine the use of model ensembling as per (Dieleman, 2014)'s approach.

References

Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU), February 2019. URL http://arxiv.org/abs/1803.08375. arXiv:1803.08375 [cs, stat].

AstroDave, AstroTom, C. R. . W. j. K. W. Galaxy zoo - the galaxy challenge, 2013. URL https://kaggle.com/competitions/galaxy-zoo-the-galaxy-challenge.

Best4. Github repository of our team. URL https://github.com/lse-st456/st456-wt2024-project-best4/tree/main.

Breiman, L. Random Forests. *Machine Learning*, 45(1): 5–32, October 2001. ISSN 1573-0565. URL https://doi.org/10.1023/A:1010933404324.

Buta, R. J. Galaxy Morphology, February 2011. URL http://arxiv.org/abs/1102.0550. arXiv:1102.0550 [astro-ph].

Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., and Liu, Z. Mobile-Former: Bridging MobileNet and Transformer, March 2022. URL http://arxiv.org/abs/2108.05895. arXiv:2108.05895 [cs].

Dai, J.-M. and Tong, J. Galaxy Morphology Classification with Deep Convolutional Neural Networks. *Astrophysics and Space Science*, 364(4):55, April 2019. ISSN 0004-640X, 1572-946X. URL http://arxiv.org/abs/1807.10406. arXiv:1807.10406 [astro-ph].

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. URL https://ieeexplore.ieee.org/document/5206848. ISSN: 1063-6919.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs].

Dieleman, S. My solution for the Galaxy Zoo challenge, April 2014. URL https://sander.ai/2014/04/05/galaxy-zoo.html.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, June 2021. URL http://arxiv.org/abs/2010.11929. arXiv:2010.11929 [cs].

Fisher, R., Perkins, S., Walker, A., and Wolfart, E. Geometric Operations - Affine Transformation. URL https://homepages.inf.ed.ac.uk/rbf/HIPR2/affine.htm.

Freund, Y. and Schapire, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997. ISSN 0022-0000. URL https://www.sciencedirect.com/science/article/pii/S002200009791504X.

Guruprasad, A. Galaxy Classification: A machine learning approach for classifying shapes using numerical data, November 2023. URL http://arxiv.org/abs/2312.00184. arXiv:2312.00184 [cs, stat].

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv:1512.03385 [cs].

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs), June 2023. URL http://arxiv.org/abs/1606.08415. arXiv:1606.08415 [cs].

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Lin, J. Y.-Y., Liao, S.-M., Huang, H.-J., Kuo, W.-T., and Ou, O. H.-M. Galaxy Morphological Classification with Efficient Vision Transformer, February 2022. URL http://arxiv.org/abs/2110.01024. arXiv:2110.01024 [astro-ph].

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A ConvNet for the 2020s, March 2022. URL http://arxiv.org/abs/2201.03545. arXiv:2201.03545 [cs].

Salama, K. Image classification with vision transformer, January 2021. URL https://keras.io/examples/vision/image_classification_with_vision_transformer/.

Schneider, J., Stenning, D. C., and Elliott, L. T. Efficient galaxy classification through pretraining. *Frontiers in Astronomy and Space Sciences*, 10, August 2023. ISSN 2296-987X. URL

https://www.frontiersin.org/articles/10.3389/fspas.2023.1197358. Publisher: Frontiers.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision*, 2017. URL https://arxiv.org/abs/1610.02391. arXiv:1610.02391 [cs.CV].

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. URL http://arxiv.org/abs/1409.1556. arXiv:1409.1556 [cs].

Tinsley, B. M. Evolution of Stars and Gas in Galaxies. 2022. URL http://arxiv.org/abs/2203.02041. arXiv:2203.02041 [astro-ph].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need, August 2023. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762 [cs].

Walmsley, M., Smith, L., Lintott, C., Gal, Y., Bamford, S., Dickinson, H., Fortson, L., Kruk, S., Masters, K., Scarlata, C., Simmons, B., Smethurst, R., and Wright, D. Galaxy zoo: Probabilistic morphology through bayesian cnns and active learning. *Journal Name*, 2020. URL https://arxiv.org/abs/1905.07424. arXiv:1905.07424 [astro-ph.GA].

Willett, K. W., Lintott, C. J., Bamford, S. P., Masters, K. L., Simmons, B. D., Casteels, K. R. V., Edmondson, E. M., Fortson, L. F., Kaviraj, S., Keel, W. C., Melvin, T., Nichol, R. C., Raddick, M. J., Schawinski, K., Simpson, R. J., Skibba, R. A., Smith, A. M., and Thomas, D. Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, November 2013. ISSN 1365-2966, 0035-8711. URL http://arxiv.org/abs/1308.3496. arXiv:1308.3496 [astro-ph].

Xia, P., Tahara, T., Kakue, T., Awatsuji, Y., Nishio, K., Ura, S., Kubota, T., and Matoba, O. Performance comparison of bilinear interpolation, bicubic interpolation, and B-spline interpolation in parallel phase-shifting digital holography. *Optical Review*, 20(2):193–197, March 2013. ISSN 1349-9432. URL https://doi.org/10.1007/s10043-013-0033-2.

ChatGPT was used in the process of writing this report. The text file containing the chatlogs is located in https://github.com/lse-st456/st456-wt2024-project-best4/blob/main/ChatGPT%20log.txt

A. Appendix

A.1. Galaxy Zoo Questions

1. Is the object a smooth galaxy, a galaxy with features/disk or a star? *3 responses*
2. Is it edge-on? *2 responses*
3. Is there a bar? *2 responses*
4. Is there a spiral pattern? *2 responses*
5. How prominent is the central bulge? *4 responses*
6. Is there anything "odd" about the galaxy? *2 responses*
7. How round is the smooth galaxy? *3 responses*
8. What is the odd feature? *7 responses*
9. What shape is the bulge in the edge-on galaxy? *3 responses*
10. How tightly wound are the spiral arms? *3 responses*
11. How many spiral arms are there? *6 responses*

A.2. Constraints on Probabilities in the Output Vector

Let p_i denote the probability value in the i^{th} position of the output vector.

1. $p_1 + p_2 + p_3 - 1 = 0$
2. $p_{12} + p_{13} - 1 = 0$
3. $p_4 + p_5 - p_2 = 0$
4. $p_{16} + p_{17} + p_{18} - p_{11} = 0$
5. $p_{19} + p_{20} + p_{21} + p_{22} + p_{23} + p_{24} + p_{25} - p_{14} = 0$
6. $p_{26} + p_{27} + p_{28} - p_4 = 0$
7. $p_6 + p_7 - p_5 = 0$
8. $p_8 + p_9 - p_5 = 0$
9. $p_{29} + p_{30} + p_{31} - p_8 = 0$
10. $p_{10} + p_{11} + p_{12} + p_{13} - p_8 - p_9 = 0$
11. $p_{32} + p_{33} + p_{34} + p_{35} + p_{36} + p_{37} - p_8 = 0$

A.3. Additional Figures

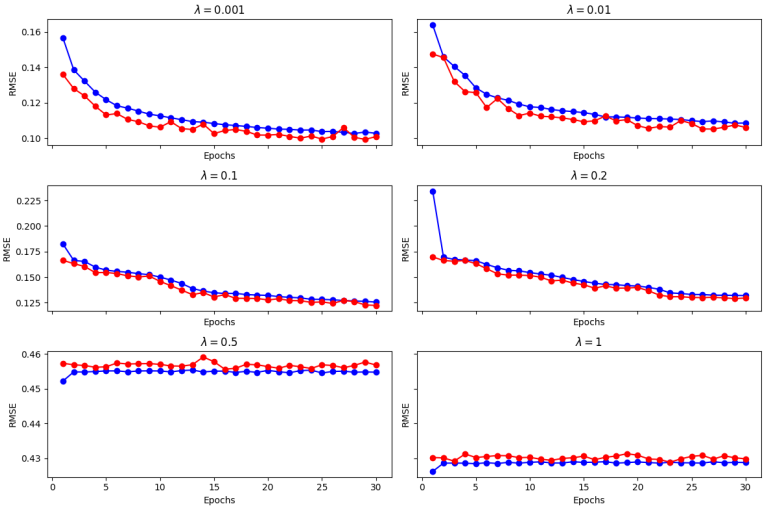

Figure 7. Training history for different values of λ in custom loss function

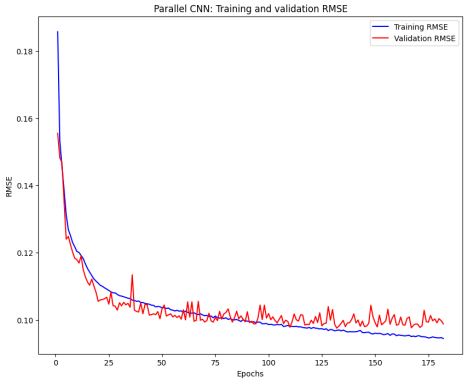

Figure 8. Parallel CNN training history

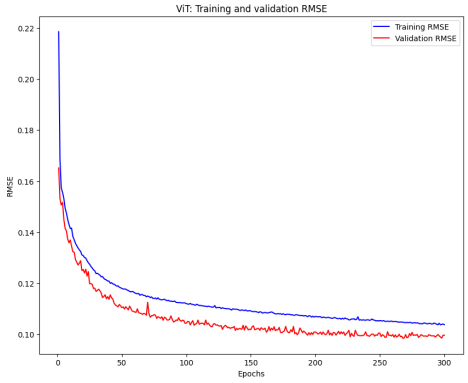

Figure 9. ViT training history

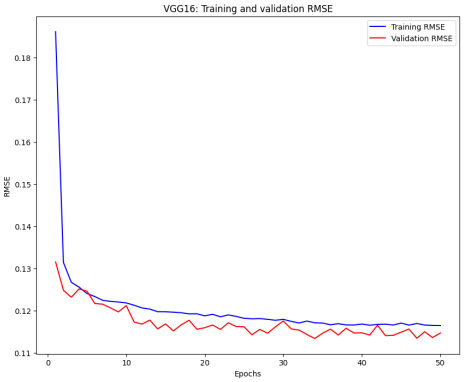

Figure 10. Fine-tuned VGG16 training history

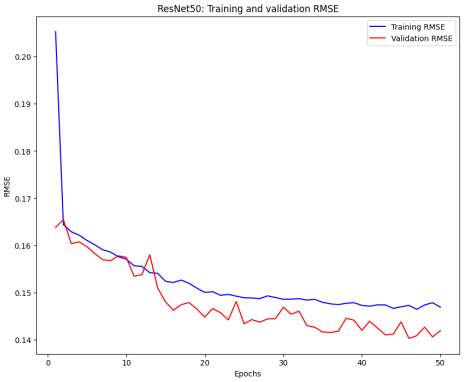

Figure 11. Fine-tuned ResNet50 training history

A.4. Array size calculation

```
# https://github.com/lse-st456/st456-wt2024-project-
best4/blob/main/scripts/array_size_calc.py
original_sample , _ = build_dataset(orig_path, n=12, preprocess=False)

byte_count = original_sample.nbytes
size = 61578 * (byte_count//12)
print(f"Linearly interpolated size: {np.round(size/1e9, 2)}")
```

Individual Contributions

1. Candidate 30855: Contributed 25% towards the final project. This candidate was responsible for the data pre-processing and data augmentation steps.
   Files (Best4):
   - p0_unzip_image_files.py
   - p1_partition_image_files.py
   - p2_build_dataset.py
   - p3_image_check.py
2. Candidate 26684: Contributed 25% towards the final project. This candidate proposed and developed the architecture of the Parallel Path Convolutional Neural Network (CNN) and contributed to model explainability using Grad-CAM.
   Files:
   - Parallel_CNN.ipynb
3. Candidate 33770: Contributed 25% towards the final project. This candidate introduced the concept of the Vision Transformer and contributed to formulating the custom loss function.
   Files:
   - VisionTransformer_B16.ipynb
   - Constraints_Test.ipynb
4. Candidate 26527: Contributed 25% towards the final project. This candidate was responsible for training the pre-trained architectures, including VGG16 and ResNet.
   Files:
   - Pretrained_Resnet.ipynb
   - Pretrained_VGG16.ipynb