DS340 Final Project Report

1. Introduction
2. Methodology
   1. Data
      1. About the Data
      2. Data Augmentation
   2. Model 1: Simple Convolutional Neural Network
   3. Model 2: Transfer Learning Model
   4. Visualisation using Heatmap
3. Results
4. Conclusions
5. References

Image Classification using Perry the Platypus dataset

Authors: Rebecca Wam U43798595, Megan Michael U03068535

# 1. Introduction

Image classification is an important task in the field of computer vision and has a wide range of applications. In this report, we explore the task of image classification using the Perry the Platypus dataset from Kaggle.

The dataset comprises images of Perry the Platypus, a well-known figure from the Phineas and Ferb animated television series, in a variety of settings and positions. Our goal is to create a model for image classification that can reliably determine whether the Perry is acting as Perry the Platypus or not.

We use two separate models to accomplish this: a simple Convolutional Neural Network (CNN) model and a transfer learning model built on the VGG16 architecture. The CNN model is a fundamental deep-learning model that makes use of numerous convolutional layers to learn the characteristics of the images. The transfer learning model, on the other hand, fine-tunes a pre-trained model, VGG16, which has previously learned the features of a large dataset, to the Perry the Platypus dataset.

Through this report, we aim to compare the performance of both models and determine which one is more effective for the task of image classification on the Perry the Platypus dataset. We also look into understanding the workings of the layers of the model through the use of heatmaps to identify the distinguishing factors in the image for classification.

# 2. Methodology

## 2.1. Data

### 2.1.1. About the Data

We decided to use the “Platypus or Perry the Platypus” dataset from Kaggle which was inspired by the famous cartoon "Phineas and Ferb" and how Dr Heinz Doofenshmirtz fails to identify Perry the Platypus until he wears his hat. The dataset is pretty small with 47 unique images in the training and validation set and 17 images in the test set. The test data set was held out and only used to test the model once training is completed to obtain the testing accuracy of our models.

### 2.1.2. Data Augmentation

Due to the small dataset, we decided to use data augmentation to help expand the dataset and boost the precision of the image classification models. In data augmentation, the original photos are subjected to a variety of transformations, including rotating, flipping, and zooming, to produce new images that are marginally different from the originals. For our data augmentation we used shear, zoom and horizontal flips.

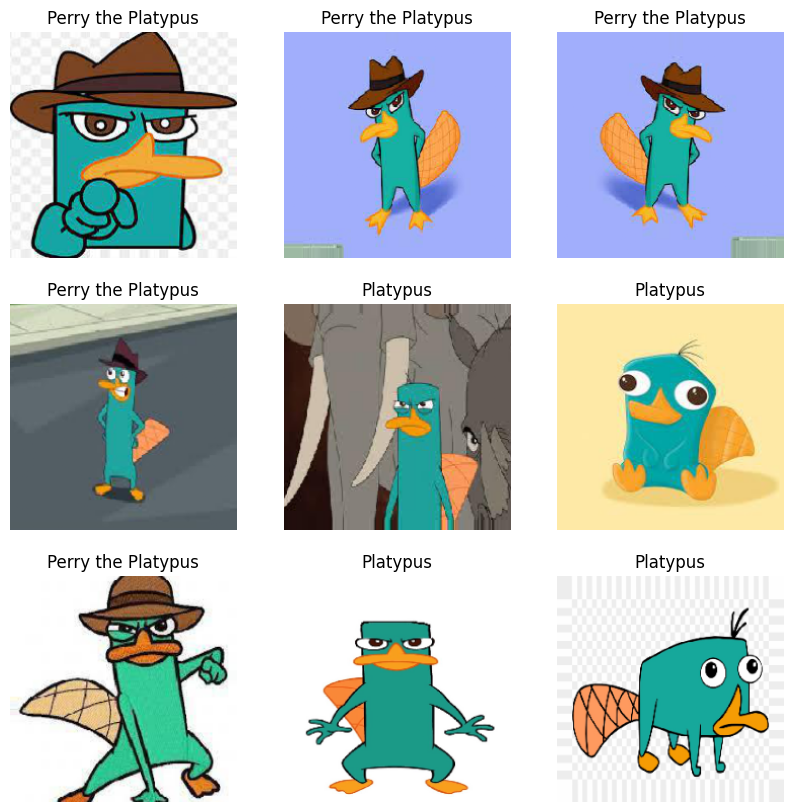


Figure 1. Perry the Platypus Augmented Training Dataset

The machine learning models can then be trained using these enhanced images, which can aid in lowering overfitting and enhancing generalization capabilities. This would help increase the number of inputs into the training models and improve their performance.

## 2.2. Model 1: Simple Convolutional Neural Network

The deep learning algorithm chosen was Convolutional Neural Network (CNN). CNN is a specialized class of artificial neural network (ANN) that uses the convolution operation in at least one of the hidden layers. CNN is fundamental to computer vision and is most popularly used for image analysis. CNN processes pixel data to detect patterns in the image to carry out automatic feature selection. This is especially suitable for our project as it is one of the best methods to analyze images. In our application, a CNN is trained on a set of images that consists of when Perry is a normal platypus and when he is in his agent form; Perry the Platypus. We chose a batch size of 32 and ran 10 epochs.



Figure 2. Summary of our simple CNN model

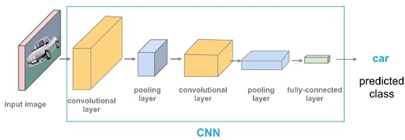


Figure 3. CNN Model Layers

**A CNN model generally has 3 main layers:**

1. Convolutional Layer: Convolution occurs in this layer. For black-and-white images, 2D convolution is used. The image is represented as a matrix of pixel data. A filter with initial randomized values is applied iteratively through the matrix and the output is stored in a feature map.

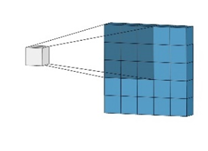


Figure 4. Visualisation of the filtering process

For coloured images, 2D convolution is used 3 times for each colour channel, namely red, green, and blue. The feature map is then passed through a Rectified Linear Unit (ReLU) activation function which is a non-linear function which allows complex relationships in the image data to be learned.

Each filter learns one set of weights through gradient descent and backpropagation. The resulting feature map encodes the features detected in the input image. The number of filters, size of the filters and stride length are hyperparameters that can be adjusted. For the first few layers of convolution, low-level features will be extracted. As the CNN gets deeper, higher-level features can be extracted through the convolution process.

1. Pooling Layer: In this layer, pooling down samples the feature map. Max-Pooling is most commonly used where another filter is iteratively applied through the feature map to extract the largest entry of the feature map within the filter. This reduces the dimensions of the layer’s output image and extracts the most relevant feature-related information from a region. Thus, the computational cost is cut, and overfitting is reduced as there are fewer parameters.
2. Fully Connected Layer: This is the final layer and it is a Multi-Layer perceptron where there are fully connected layers which connect every neuron in a layer to every neuron in another layer. The image data from the final pooling layer is flattened into a vector and passed into this layer which then outputs a classification based on the features.

**Advantages of the CNN model:**

1. Computationally efficient: With general ANN, the input images must be converted to one-dimensional vectors which exponentially increases training parameters. This is not required for CNN and furthermore, the pooling layers reduce image dimensionality, thus computational cost is greatly reduced.
2. Feature Extraction: With ANN, each feature in an image must be measured and provided as concrete data points. CNN automatically extracts thousands of such features without human supervision.
3. Translation invariant to some degree: Slight image shifting still produces similar feature maps so most important features will still be detected but at different locations of the image. Thus, CNN is robust to translated data.

## 2.3. Model 2: Transfer Learning Model

Using pre-trained models as a starting point for creating new models is a common practice in image classification. A well-known pre-trained model called the VGG16 has attained cutting-edge performance on a variety of image classification tasks. Transfer learning with the VGG16 model can be utilized in the context of the Perry the Platypus dataset to enhance a simple CNN model and achieve greater accuracy in the image classification challenge.

The VGG16 model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers, and it has been trained on the ImageNet dataset, which contains over 1 million images in 1000 categories. By using transfer learning, we can leverage the pre-trained weights of the VGG16 model and fine-tune the last few layers of the model for the specific classification task of Perry the Platypus images. The earlier convolutional layers that extract universal, low-level features such as edges can be frozen, and the later layers can be trained to extract features specific to the image classification problem.

To implement transfer learning with the VGG16 model, we use Keras, a popular deep-learning framework. We can then train the VGG16 model on the Perry the Platypus dataset, using techniques such as data augmentation to improve the performance of the model. We also chose a batch size of 32 and ran 10 epochs.

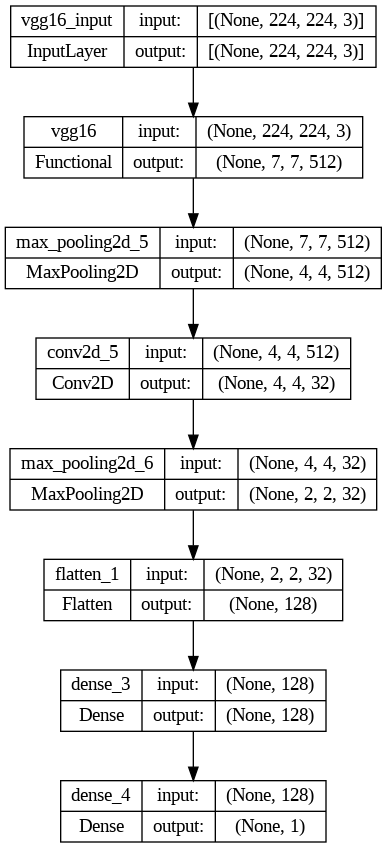


Figure 5. Summary of our transfer learning model

## 2.4. Visualisation using Heatmap

Heatmaps are a useful tool in image classification that can help to visualize which part of the input image is being used for the classification task. Heatmaps are generated by the model after the classification process, and they highlight the areas of the image that the model has deemed most relevant for the classification. In the context of the Perry the Platypus dataset, heatmaps can help to identify the specific features or characteristics of Perry's appearance that the model is using to classify the images into different classes.

To generate heatmaps, we used Grad-CAM (Gradient-weighted Class Activation Mapping), which calculates the gradient of the output with respect to the convolutional feature maps and generates a heatmap that highlights the relevant regions of the image. By visualizing the heatmaps, we can gain insights into the inner workings of the models and understand which features of Perry's appearance are most important for the classification task.

The last convolution layer is tracked and used to plot the heatmap. This is easy to implement for the simple CNN model. However, for the transfer learning model, we were only able to generate the heatmap after adding additional convolution layers. We overlayed the heatmaps on the original image of the test set and displayed the classification results above each image.

# 3. Results

We evaluated our models on the test dataset. The accuracy of the simple CNN model is around 53% which is not as good as the transfer learning model with an accuracy of around 77%. The simple CNN model would not be helpful in this classification as a 53% accuracy is almost like taking a wild guess of whether the image is of Perry the Platypus or Platypus. Using transfer learning helped to achieve higher accuracy than the simple CNN model as the simple CNN model would require more time and computational resources to train from scratch. For transfer learning, we can build upon the knowledge and insights gained from the pre-trained model and apply it to a new and specific image classification task, such as the Perry the Platypus dataset. Hence, doing transfer learning with the VGG16 model can help to achieve higher accuracy with fewer training epochs and a smaller dataset.

When implementing heatmap for the simple CNN model, we found that the heatmap shows the shape of the platypus. This signals that the model simply classifies any image with a platypus as Perry the Platypus.

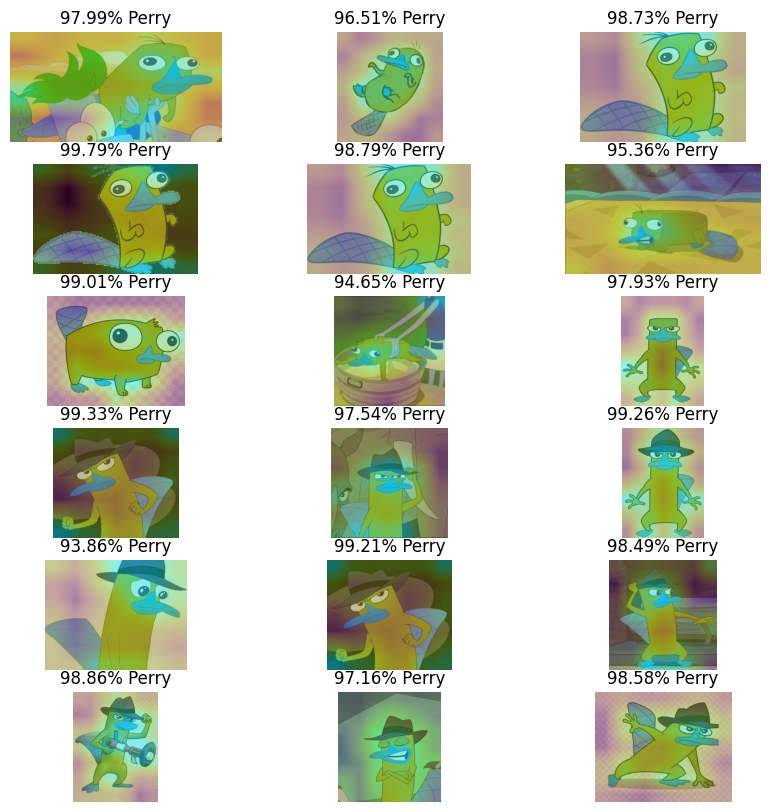


Figure 6. Heatmap for the simple CNN model

However, for the transfer learning model, it is unclear what the heatmap is picking up. This could be because as we progress deeper into the model, the feature maps show less and less detail so it is hard to interpret what it is detecting. Since the VGG16 model has many layers, by the time it reaches the last layer, the input image has been processed so many times that it becomes hard to identify what exactly the model is detecting.



Figure 7. Heatmap for the transfer learning model

# 4. Conclusions

In this project, we have explored the effectiveness of CNN models and transfer learning models, specifically using the VGG16 model, for the image classification task on the Perry the Platypus dataset from Kaggle. Furthermore, we have shown how heatmaps can be used to visualize the classification process and identify the specific features of Perry's appearance that the model is using for the classification.

Through our experiments, we have found that the transfer learning approach with the VGG16 model outperforms the simple CNN model and achieves higher accuracy on the Perry the Platypus dataset. We have also seen how heatmaps could provide valuable insights into the inner workings of the models and help to identify potential biases in the dataset or the model. We also realised that the more layers the model have, the harder it is for us to understand what it is detecting with the heatmap.

Overall, our findings suggest that transfer learning with pre-trained models such as VGG16 can be a highly effective technique for image classification tasks, particularly when dealing with relatively small datasets. Additionally, the use of heatmaps can provide a deeper understanding of the model which could then be used to help to improve the accuracy and robustness of the classification models.

**Future Improvements**

1. In our current model, we used less than 100 images before augmentation, from there we had to split the dataset into 3 categories; training, validation and testing. This left us with very few training images for the model to fully encapsulate all the features of Perry. We could increase the size of the dataset by finding more images and self-labelling them.
2. To further improve the accuracy of our models, we could use other image augmentations to increase the data count and make it more robust. Such augmentations other than zoom and shear are zca whitening, rotation and so on. Additionally, we can process the images via contrast stretching or histogram equalization to standardize the image's colouration. These image processing techniques help enhance images’ contrast by spreading out the most frequent pixel intensity or intensity range of the image. This would accentuate the features and edges within the images. This would also allow the model to read the images without noise from the brightness of the image.
3. To further improve our visualisations, we could also work on flattening the nested transfer learning model so that we could access the convolution layers in the VGG16 model and visualise the feature maps for those layers. This could give us more insights on what the model is detecting to distinguish between Platypus and Perry the Platypus.

# 5. References

AloneTogether. “How to view images in directory iterator object tensorflow.” *Stack Overflow*, 2 June 2022, https://stackoverflow.com/questions/72473986/how-to-view-images-in-directory-iterator-object-tensorflow. Accessed 1 May 2023.

Brownlee, Jason. “How to Configure Image Data Augmentation in Keras - MachineLearningMastery.com.” *Machine Learning Mastery*, 12 April 2019, https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/. Accessed 1 May 2023.

Brownlee, Jason. “How to Visualize Filters and Feature Maps in Convolutional Neural Networks - MachineLearningMastery.com.” *Machine Learning Mastery*, 6 May 2019, https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/. Accessed 1 May 2023.

Chollet, François. “Image classification from scratch.” *Keras*, 27 April 2020, https://keras.io/examples/vision/image\_classification\_from\_scratch/. Accessed 1 May 2023.

Dertat, Arden. “Applied Deep Learning - Part 4: Convolutional Neural Networks.” *Towards Data Science*, 8 November 2017, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed 1 May 2023.

Fisher, R., et al. “Contrast Stretching.” 2003, https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm. Accessed 1 May 2023.

“Histogram Equalization.” *scikit-image*, https://scikit-image.org/docs/0.9.x/auto\_examples/plot\_equalize.html. Accessed 1 May 2023.

Kadlaskar, Amruta. “Image Classification using CNN : Python Implementation.” *Analytics Vidhya*, 14 June 2021, https://www.analyticsvidhya.com/blog/2021/06/image-classification-using-convolutional-neural-network-with-python/. Accessed 1 May 2023.

Khoshgoftaar, Taghi M. “A survey on Image Data Augmentation for Deep Learning - Journal of Big Data.” *Journal of Big Data*, 6 July 2019, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0. Accessed 1 May 2023.

Kim, Suhyun. “A Beginner's Guide to Convolutional Neural Networks (CNNs).” *Towards Data Science*, 14 February 2019, https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8. Accessed 1 May 2023.

Koehrsen, Will. “Transfer Learning with Convolutional Neural Networks in PyTorch.” *Towards Data Science*, 26 November 2018, https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce. Accessed 1 May 2023.

Kyaw Saw Htoon. “A Tutorial to Histogram Equalization.” *Medium*, 18 August 2020, https://medium.com/@kyawsawhtoon/a-tutorial-to-histogram-equalization-497600f270e2. Accessed 1 May 2023.

Meel, Vidushi. “ANN and CNN: Analyzing Differences and Similarities - viso.ai.” *Viso Suite*, https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/. Accessed 1 May 2023.

Odemakinde, Elisha. “Everything about Mask R-CNN: A Beginner's Guide - viso.ai.” *Viso Suite*, https://viso.ai/deep-learning/mask-r-cnn/. Accessed 1 May 2023.

Pawar, Dipti. “Improving Performance of Convolutional Neural Network!” *Medium*, https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7. Accessed 1 May 2023.

Renotte, Nicholas. “ImageClassification/Getting Started.ipynb at main · nicknochnack/ImageClassification.” *GitHub*, https://github.com/nicknochnack/ImageClassification/blob/main/Getting%20Started.ipynb. Accessed 1 May 2023.

Rosebrock, Adrian. “Grad-CAM: Visualize class activation maps with Keras, TensorFlow, and Deep Learning.” *PyImageSearch*, 9 March 2020, https://pyimagesearch.com/2020/03/09/grad-cam-visualize-class-activation-maps-with-keras-tensorflow-and-deep-learning/. Accessed 1 May 2023.

Sehgal, Akshay. “How to force Keras VGG16 model show and include detailed layers when being used in new customized models.” *Stack Overflow*, 27 July 2021, https://stackoverflow.com/questions/68551375/how-to-force-keras-vgg16-model-show-and-include-detailed-layers-when-being-used. Accessed 1 May 2023.

Shaju, Riya Eliza. “Platypus or Perry the platypus.” *Kaggle*, https://www.kaggle.com/datasets/riyaelizashaju/platypus-or-perry-the-platypus. Accessed 1 May 2023.