A Comparison of Pre-Trained Models Accuracy for Cat/Dog Image Classification using Transfer Learning

Bolanle Oladeji

Department of Computer Science, The College of Wooster

May 8, 2022

Contents

1 Abstract

With the rise of many image-capturing systems, the ability to classify vast amounts of unstructured data is extremely relevant. Over the past few years, Convolutional Neural Networks have emerged as the best tool for image classification due to their ability to easily extract features and high accuracy. A number of pre-trained models– CNN models that have been trained on large datasets –exist; and through the process of transfer learning, be easily repurposed for a number of problems. The question of which of these models is the most accurate then arises. For my Junior Independent Study Thesis, I compared the accuracies of three fine-tuned pre-trained models for cat/dog image classification. Implementing transfer learning on the models yielded very high test accuracies. Another discovery was there was little to no difference in the models' accuracies when fine-tuned with the same hyperparameter tuning and epochs. This just highlights how powerful transfer learning can be especially in the realm of classifying similar tasks to the original models'. A Graphical User Interface was also implemented to test the most accurate of the models on any cat/dog image.

2 Introduction

Convolutional Neural Networks are often made of many layers that require hours of training time. Transfer Learning is the reuse of a previously learned model for a new task [3]. Transfer Learning can help in reducing training time as well as improve the performance of a neural network. Tech giants like Google and Microsoft have utilized their resources to train models on large datasets. These pre-trained models often have the ability to classify images into hundreds and sometimes thousands of categories.

Image Classification is under the greater field of computer vision and is the process of identifying and grouping images according to specific labels. Image Classification is particularly useful in the medical research field, autonomous vehicle industry and in object identification

in traffic management systems. For example, in the autonomous vehicle industry, using captured imagery from sensory input devices, cars are able to detect pedestrians and traffic signs and classify them into their appropriate categories. Deep learning techniques as opposed to traditional image processing is preferred, because of its greater accuracy and speed.

The use of transfer learning on pre-trained models is very useful in image classification tasks. The aim of this paper is to explore 3 types of pre-trained models and compare their accuracies after transfer learning is implemented on cat/dog images. But first, we must explore the theoretical background behind these concepts.

3 Background

3.1 Machine Learning

Machine Learning is a sub-field of artificial intelligence, that is concerned with programming computers to become adept at imitating human behavior by using example data or past experience. To successfully train a system, we define mathematical models according to some parameters. The model could either be descriptive, that is to gain knowledge from data, predictive, to make predictions about the future or both [2]. Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions. There are two central approaches to statistics: frequentist estimators and Bayesian inference [4].

1. **Frequentist Estimators:** Frequentist estimators is the branch of inferential statistics whose underlying principle is based on the repeating of experiments, that is the frequency in long running experiments and using the data from the current experiment to make predictions. The more we repeat experiments, the more accurate our estimates should be. Any uncertainty in probabilistic estimates is chalked up to be sampling er-

ror and theoretical differences. P-values and confidence intervals are examples of the frequentist estimator approach [4].

2. **Bayesian Inference:** This is based on Bayes' Theorem. In this, past knowledge of similar experiments is incorporated and combined with the data from the current experiment to make predictions. An application of this is in Bayesian Linear Regression [4].

There are four major ways that Machine Learning systems can be categorized based on the way in which they are supervised. These are: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning and Reinforcement Learning. Of these four, Supervised Learning is the most important that we shall discuss, as it is used in image classification.

3.1.1 Supervised Learning

Supervised learning can be described as the subset of machine learning dedicated to finding relationships between characteristics of the data that have already been measured [8]. In this, we have input and output variables and use the algorithm to learn the mapping function from the input to the output. The training set includes the solutions which are called labels. Supervised learning can be used in classification problems or regression problems. This will be the learning technique that is extensively explored in this paper. Applications of supervised learning can be found in linear and logistic regression as well as in neural networks [5].

3.1.2 Unsupervised Learning

Unsupervised learning can be thought of as finding relationships between things you have measured and things that have not been measured yet [8]. In this type, there are no labels in the training set, there is only the input variable(data). The algorithm tries to model the data on to see any underlying patterns on its own. Visualization algorithms are good exam-

ples because the machine tries to detect patterns that the unlabelled forms. Unsupervised learning algorithms include clustering, association rule learning and anomaly detection.

3.1.3 Semi-Supervised Learning

In this case, there are few labelled data and many unlabelled instances. Systems like this are usually trained in an unsupervised manner and then fine-tuned using supervised learning. The model learns from the few examples of labelled data and then makes predictions on the unlabelled ones. Semi-supervised learning is usually employed in situations when labelling data is either too expensive or too challenging. A good example of a semi-supervised learning algorithm are deep belief networks [5].

3.1.4 Reinforcement Learning

Reinforcement Learning is a subset of machine learning that enables an agent, the learning system, to observe the environment, select and perform actions and get rewards in return. Rewards can sometimes be negative in order to punish unwanted behavior by the agent. The agent, in turn must learn by itself the best solution to get the maximum reward. A great example of reinforcement learning algorithms is in AI gaming systems like DeepMind's AlphaGo that beat the world champion after learning the best policy by analyzing and playing many games of Go [5].

3.1.5 Applications of Machine Learning

- **Learning Associations**

  Machine Learning can be used to find relations or associations between variables in data. The algorithm tries to find associations that take place more frequently than others [4]. An application of this can be found in market basket analysis, where large retailers try to find associations between items that they sell.

- **Classification**

  Machine Learning can also be used to classify an input into a number of categories. Classification uses supervised learning to achieve its goals. Deep learning is best suited for this and common applications can be found in facial recognition, bio-metrics and in image classification.

- **Regression**

  This also utilizes Supervised Learning. The model tries to predict a numerical value based on its input [4]. In this, we predict a target value based on a set of features. An application of this is in the insurance industry, where companies try to predict claim amounts.

- **Transcription**

  The machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form.[4] An application of transcription is in voice to text conversion.

3.2 Deep Learning and Artificial Neural Networks

An Artificial Neural Network (ANN) is a Machine Learning model inspired by the networks of biological neurons found in our brains. They were introduced in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts [5]. Networks represent the composition of many different functions. Functions are often connected in a chain and the length of the chain gives the depth of a model [4]. Deep learning models are represented by series of operations that have at least two, nonconsecutive nonlinear functions involved. Layers are a series of linear operations followed by a nonlinear operation. Deep Learning

therefore, is a subset of machine learning that is based on artificial neural networks. They are neural networks with more than one hidden layer [8].

3.2.1 The Biological Neuron

An artificial neuron operates much like the biological neuron. Biological neurons produce electrical impulses called action potentials. These travel along the axons, which are long thread-like parts that help to conduct impulses along a neuron. The impulses make the synapses release chemical signals called neurotransmitters. When a neuron receives enough of these neurotransmitters (only when the strength of the signals exceeds a certain threshold), it fires its own electrical impulses. Whether a neurotransmitter actually fires is dependent on the exact type, some neurotransmitters actually inhibit neurons from firing. The output signal from one neuron can form the input signal for another neuron. Neurons are organized in a vast network of billions, connected to each other and are able to compute highly complex actions. Thus, a Biological Neuron is a threshold logic unit (TLU). A TLU is an object that has numbers as its inputs and outputs, with each input unit weighted. The TLU then computes a weighted sum, and if that sum meets or exceeds a threshold, outputs some number [5].

3.2.2 The Perceptron

The Perceptron was invented in 1957 by Frank Rosenblatt. It can be used to classify data into two parts or simple binary classification. Because it is one of the simplest ANNs, we can study its structure to get a general overview of how neural networks work. The perceptron consists of a single neuron and functions conceptually like the biological neuron [3]. Unlike biological neurons, artificial neurons do not fire electrical impulses, they instead emit continuous signals [5]. The artificial neuron calculates the weighted sum of the inputs to represent the total strength of the input signals. It considers a certain threshold value and then determines whether to fire the output 1 if the signal exceeds the threshold or 0 if

the signal does not exceed the threshold.

Each input signal is assigned a weight based on how much influence it has on the output. The weight represents the strength of connection between the units. Each artificial neuron then calculates a weighted sum of all incoming inputs by multiplying the signal with the weights and adding a bias. The weighted sum then goes through a function called an activation function or transfer function to decide whether the neuron should be fired or not, which results in output signals for the next level [3].

The weighted sum is simply the sum of all the inputs multiplied by their weights and added to a bias. A bias is essentially an extra input node to the neurons that always has a value of 1; the bias can be learned, adjusted and ensures that regardless of the input values, an activation still occurs in the neuron. The activation function is used to determine the output of the neuron and takes the weighted sum and activates the neuron depending on if it is higher than a certain threshold. In the case of the Perceptron, the step function which produces a binary output is used.

3.2.3 The Multilayer Perceptron (MLP)

A Multi Layer Perceptron is simply composed of an input layer, output layer and a few hidden layers. Neural networks consist of three layers: the input layer, the hidden layer(s) and the output layer. Input neurons form the input layer and this is where information is inputted into the network. The hidden layer is any layer that is between the input layer and the output layer.The layers that are close to the input layer are called the lower layers, and the ones that are close to the outputs are called the upper layers. Multilayer Perceptrons are especially important when it comes to non-linear data (data that cannot be split by one straight line) [3]. And so, we stack neurons on top of one another in a hidden layer. When all the neurons in a layer are connected to every neuron in the previous layer, the layer is called a fully connected layer, or a dense layer. When an ANN contains a lot of deep hidden

layers, it is called a deep neural network (DNN).

3.2.4 Activation Functions

Activation Functions help in introducing non-linearity in our network. There are different types of activation functions that include:

1. **Linear Activation Function**: The input passes through the function unchanged; that is, it is not transformed into a non-linear function. Here, the activation is proportional to the input. The problem with this is that no matter how many layers are added, the last layer will still be a linear function of the first. This means that the neural network will essentially be just one layer. Also, because the derivation of a linear function is constant, every time backpropagation is carried out, the gradient remains the same [3].

   activation(f) : $f(x) = wx + b$ where w is the weight and b the bias value.

2. **Heavyside or Binary Step Function** This produces a binary output. If the input is > 0, the neuron is activated, otherwise it is not. It is mainly used for binary classification problems, hence it cannot be used for multi-class classification [3].

3. **Sigmoid/Logistic Function** This function predicts the probability of a class in the case of binary classification. The sigmoid function has output values that are in the range 0 to 1.

   sigmoid(f) : $f(x) = 1/(1 + e^{-x})$

4. **SoftMax Function** The SoftMax function is a combination of multiple sigmoid functions; it returns the probability of each class in a multi-class classification problem. The function also has output values that are in the range 0 to 1 and all class' probabilities sum up to 1.

5. **Hyperbolic Tangent Function (tanh)** The tanh function is very similar to the sigmoid, except that its ouput values are in the range -1 to 1. The function has the effect of centering its output values around zero unlike the sigmoid, which helps the next layer in learning.

6. **Rectified Linear Unit (ReLU** The rectified linear unit (ReLU) activation function will only activate a neuron if the output is above zero. If the output of the linear transformation is less than zero, the neuron will be deactivated. One problem with ReLU is that when x is negtaive, the derivative is 0 which affects the backpropagation process. The Leaky ReLU function helps to fix this by introducing a small positive slope when x is negative [3].

   ReLU (f) : $f(x) = max(0, x)$

3.2.5 Neural Network Training Process

Training a neural network involves:

Feedforward Process and Feature Learning:

The process of computing the linear combination and applying the activation function is known as the Feedforward process. In this, the information flows in a forward direction from the input layers to the output layer. After the feedforward process comes feature learning, here, the network learns the representations needed for feature detection or classification. The network learns patterns from one layer and these features are transformed into new features with the next layer and so on.

Error Functions: The error function is a method of obtaining how far off the neural network's predictions are from the actual results. If our model has a great loss then we need to train it further to minimize the loss. There are two major categories for loss functions: Regression Losses and Classification Losses. Mean Squared Error is commonly used for regression while cross-entropy is used for classification. In the mean square error, the difference between the predicted and actual values is squared and averaged ensuring that

the error is always positive. With Cross-entropy, it measures the difference between the predicted probability distribution and the actual probability distribution [3].

Optimization Algorithms: After finding the error function, we will need to optimize it so that we have the minimum error. Usually, parameters such as the weight and learning rates are changed in order to reduce losses. There are different types of Optimization algorithms such as Gradient Descent, Adaptive Gradient and Adam. We will only discuss Gradient Descent and Adam in line with the purposes of this paper.

1. **Gradient Descent:** Gradient Descent has several variations which include stochastic gradient descent, batch gradient descent and mini-batch GD (MB-GD). It is an optimization algorithm used to find the local minimum/maximum of a differentiable function. Gradient descent works by taking "steps" down the curve to get to the local minima. To do this, we need the step size (learning rate) and the step direction (gradient) [3]. To find the gradient, we take the derivative of the error with respect to the weight.

   The learning rate, which constitutes the step size is one the most important hyperparameters; a larger one implies faster learning and vice versa. Hyperparameters are simply parameters that are tunable. Smaller learning rates are preferred because even though they take longer, they eventually lead to the minima. A large one can cause the error to oscillate, meaning that the network never descends.

2. **Adaptive Moment Estimation (Adam):** Adam is one of the best optimizers because it helps train a neural network model more more quickly than the others. Adam does this by taking into account the exponentially decaying average of gradients [3].

Backpropagation: After optimizing the error function, we get a change in weight that would minimise error. We then use back-propagation to updates the weights across the network. It is an iterative process that involves propagating derivatives of the error with

respect to each specific weight. It starts at the end of the network and then recursively computes gradients up to the input layer and then updates the weights [3].

3.2.6 Types of Artificial Networks

There are many types of Artificial Neural Networks including the FeedForward Neural Networks, Regulatory Feedback Networks, autoencoders, Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). CNNs will be discussed in an in-depth manner in the Computer Vision section.

1. **FeedForward Neural Networks**

   These are also called Multilayer Perceptrons (MLPs). An MLP consists of an input layer, many hidden layers and an output layer. Feedforward networks are used to approximate some function. The function is evaluated from x and then gives an output y. Feedforward networks do not have feedback connections meaning that the output is not fed back into the model itself. Instead, the information flows through the function to give an output y [5].

2. **Deep Belief Networks**

   Deep Belief Networks are based on Restricted Boltzmann Machines (RBMs). Restricted Boltzmann Machines are two-layered artificial neural networks with all of the neurons connected to each other. It is restricted in the sense that there can be no connections between neurons in the input layer and also between neurons in the hidden layer. There is no ouput layer with RBMs. When we stack RBMs on top of each other, we get a Deep Belief Network. DBNs have the incredible generating capacity that allow it to be used in image recognition and in general unsupervised learning tasks [5].

3. **Recurrent Neural Networks**

A recurrent neural network is designed just like a Feed forward neural network except that it has feedback connections. This means that it has connections that point backward, thus it has an infinite impulse response [5]. RNNs are commonly used in natural language processing such as in handwriting recognition and speech recognition.

4. **Autoencoders**

Autoencoders are artificial neural networks that use an unsupervised learning technique to compress dense representations of the input data, called latent representations or codings. The autoencoder then learns how to decode the data from the compressed form. Autoencoders are commonly used in anomaly detection and even the generation of image data. [5]

3.3 Computer Vision and Image Classification

Computer vision is concerned with the visual perception part: it is the science of perceiving and understanding the world through images and videos by constructing a physical model of the world so that an AI system can then take appropriate actions [3]. Image Classification falls under the broad field of computer vision. It is the labelling of images into their predefined set of categories. There are many different algorithms such as the K-Nearest Neighbor Algorithm and the Multi-layer Perceptron that can be used for classification. Convolutional Neural Networks are especially good in computer vision image classification tasks and have been known to achieve extremely high accuracy in complex visual tasks.

3.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are used in the field of Computer Vision especially in image classification and object detection. What separates CNNs for deep neural networks is the addition of three new types of layers, the convolutional layers, pooling layers and fully connected layers. Convolution is a mathematical concept of combining two functions

to produce a modified function. CNNs have two components: the feature extraction part and the classification part.

In feature extraction, the convolutional and pooling layers are involved in a set of operations to extract features from the data. The convolution layer uses user-specified kernels or convolution filter to slide over the input. A matrix multiplication is then performed pixel by pixel to form a feature map.

The convolutional layer takes five arguments: filters, kernel size, activation function, stride and padding. Kernel size is simply the size of the convolutional filter matrix (e.g. 3 x 3) while filters specifies the number of convolutional filters in each layer. Stride is how much the filter slides over the input and padding is simply adding zeros around the border of an image input to ensure uniformity in image width and height. The pooling layer's goal is to reduce the size of the input image in order to reduce computational load and prevent over fitting [5]. After feature extraction, the fully connected layers are responsible for classification. These layers have neurons that are fully connected to the cativations in the previous layer. Because they can only accept 1-dimensional data, image data has to be flattened.

Overfitting Problem: There are two things that could happen after training a model; it could be underfitted or overfitted. Underfitting is when the model does not fit the data while overfitting occurs when the model fails to learn the specific features, only fits the training data and is not generalizable to other data [4]. To reduce the risk of overfitting, a dropout layer is usually added after flattening. Dropout simply turns off a certain percentage of neurons in a layer in the network.

Dataset Preprocessing: Before training a model, it is important to split our data into train dataset, testing dataset and sometimes validation dataset. A training dataset is used to train the model so that it learns. The testing dataset is used for evaluating the model after training is complete. We use the validation dataset to fine tune parameters during training.

Image Preprocessing: There are different types of preprocessing techniques that may be applied in order to get your image data ready for training. These include normalization, data augmentation and image resizing. Image resizing in particular, is important because neural networks require all images to be of the same size. Data augmentation involves applying image techniques to an image in order to create multiple slightly different copies to supplement the number of data available. Data normalization involves rescaling pixel values to fall within a certain range of say, 0 to 1 to improve performance.

3.3.2 Transfer Learning

Transfer Learning is the transfer of the feature maps gained from a task by a model to a new task [3]. This involves using a pre-trained model as the starting point for a model on a new task. A pre-trained model is a network that has been previously trained on a large dataset. To implement transfer learning, we download the pre-trained model with its pre-trained weights, then we remove the classification layers and add our own. After this, we freeze its feature extraction layers and add our own classification dense layer. Freezing simply means freezing trained layers to prevent them from being re-trained. After doing this, we build a new model whose input is the input of the base model and output is the output of the last activation function layer. This new model is comprised of the convolutional and pooling layers of our base model as well as the untrained classification layer.

4 In Practice

After going through the theoretical framework behind image classification, we can now implement our own CNN models using transfer learning. For this, I will be focused on cat/dog image classification because it is one of the most widely explored classification problems. The algorithms were implemented using the Tensorflow Keras API framework [1]. The dataset being used is an excerpted version from the popular "cat versus dogs" dataset from the Play-

ground Prediction Competition on Kaggle. It contains 3000 images: 2000 for the validation set and 1000 for the training set. In addition to this, 20 percent of the validation batches formed the test dataset.

The aim of my project is to implement transfer learning using three different pre-trained models and compare their test accuracy and losses. The three models are the MobileNet-v2, Inceptionv3 and the ResNet50 models.

MobileNet-v2: MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It was developed by researchers at Google. It has 53 Convolutional layers and 1 average pooling layer. The model was trained on more than 1 million images from ImageNet and it can classify into more than 1000 categories. ImageNet is a large visual database with more than 14 million images hand-annotated.

Inceptionv3: Inceptionv3 is a convolutional neural network that is 42 layers deep. It was also trained on more than 1 million images from ImageNet. It can classify into 1000 object categories It also has lower error rates in comparison to its contemporaries. There are parallel layers instead of deep layers so the model is wider rather than deeper [3].

ResNet50: The Residual Network was developed by researchers at Microsoft. ResNet was particularly novel because it was able to overcome the vanishing gradient problem. CNNs commonly have the Vanishing Gradient problem which occurs when as more layers are added, the gradient of the loss function approaches zero which makes it hard to train the network. ResNet was able to overcome this because Residual Networks utilize something called a skip connection which is a direct connection that skips over some layers of the model. The output is not the same due to this skip connection. ResNet50, in particular, is a convolutional neural network architecture that is 50 layers deep. It works by stacking residual blocks on top of each other. Each layer feeds into the next layer and then directly into layers that are 2-3 hops away [3].

Image Pre-Processing:

Image Resizing: Images were resized to 160 by 160.

Image Augmentation: adding slightly modified copies of existing data (flipped some images).

```
data_augmentation = tf.keras.Sequential([
tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
])
```

Finally, standard pre-processing from the models was also used.

Initially, all the models were compiled and optimized with Adam. They had the follow initial accuracies and losses.

Model	Initial Accuracy	Initial Loss
Mobilenet-v2	0.4400	0.8500
ResNet50	0.5260	0.9627
Inceptionv3	0.5300	0.9200

After downloading the models and compiling them with their pre-trained weights; the lower layers were frozen and a dense layer with one hidden unit was added. After this, I trained the model for 10 epochs. An epoch is a training iteration when the model goes a full cycle and sees the entire training dataset at once [3]. Then, I began the transfer learning process using the fine-tuning approach. I unfroze all of the layers and then froze all the layers before the 100th layer. The model was then compiled again at a very low base learning rate of 0.001 to reduce the risk of overfitting. Then it was trained for an additional 10 epochs. Out of all the models, MobileNetV2 took the shortest amount of time to train, followed by the ResNet50 and then the Inceptionv3 model.

```
RN5_model.trainable = True
# Fine-tune from this layer onwards
```

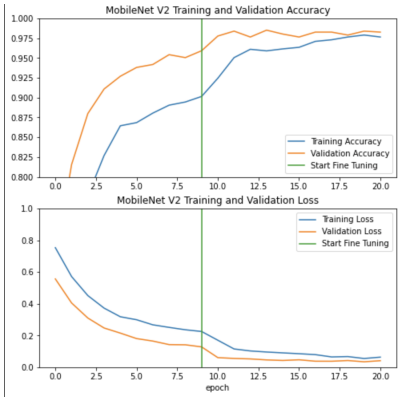

Figure 1: MobileNetV2 model

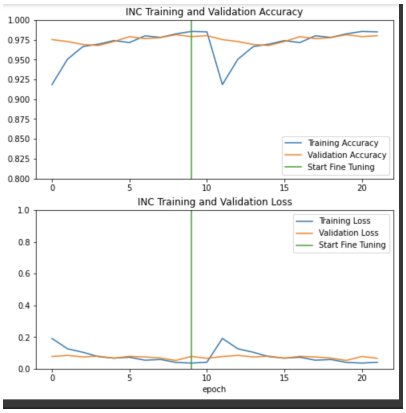

Figure 2: Inceptionv3 model

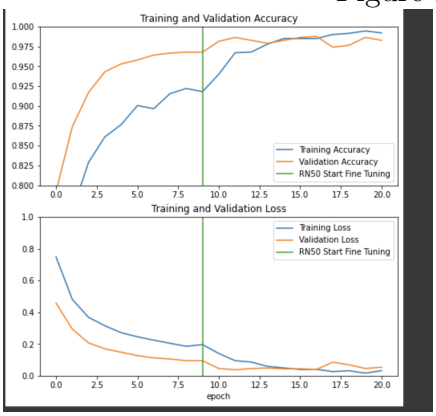

Figure 3: ResNet50 model

```
fine_tune_at = 100
# Freeze all the layers before the ‘fine_tune_at‘ layer
for layer in RN5_model.layers[:fine_tune_at]:
    layer.trainable =  False
```

Figures 1, 2 and 3 show accuracy increasing and loss decreasing with the epoch training. After fine-tuning we see the accuracy and loss decrease for the models. There was not much of a difference in the Test Accuracy as they all achieved very high levels of test accuracy after fine-tuning. Wang, Lee and Chang [7] compared 4 fine-tuned applications and were able to achieve very high levels of accuracy with relatively similar results. I chose the ResNet50 model for my Graphical User Interface Implementation because it had the relatively highest test accuracy.

The testing loss and accuracy for all three models is presented in the following table:

Model	Test Accuracy	Test Loss
Mobilenet-v2	0.984375	0.026739
ResNet50	0.994791	0.024535
Inceptionv3	0.989583	0.049519 %

Graphical User Interface (GUI):

The GUI for this project was implemented using Gradio. Gradio is an open-source python library used for making user interfaces for machine learning projects [6]. The interface accepts image as the input and then outputs the model's prediction for that image via text.

5 Limitations and Future Work

Due to limited time and resources , the models were only trained and fine tuned for a total of 20 epochs. The same optimization algorithm was also used for all 3 models, a comparison of algorithms might yield more significant results. Future work will consist of a higher

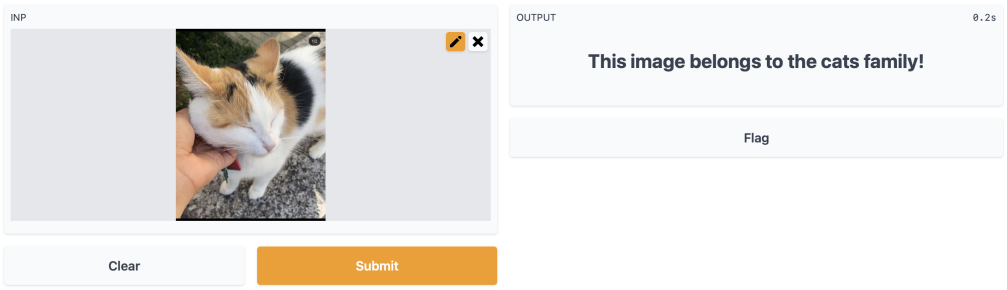

Figure 4: Graphical User Interface

number of epochs and a more varied hyperparameter tuning. Also, early stopping could be implemented in future work to prevent over training of the data. The Graphical User Interface could also be improved to allow users to select which of the models they want to use in predicting their image.

6 Conclusion

This paper focused on the comparison of the prediction accuracy of three CNN models of ResNet50, Mobilenet-v2 and Inceptionv3. From the results of the analysis, it is clear that transfer learning greatly increases a model's accuracy. The results also show that the Inception v3 model has the highest accuracy among all 3 models.

References

[1] Dogs vs. cats plaground prediction competition.

[2] E. Alpaydin. *Introduction to Machine Learning, fourth edition*. Adaptive Computation and Machine Learning series. MIT Press, 2020.

[3] M. Elgendy. *Deep Learning for Vision Systems*. Manning Publications, 2020.

[4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[5] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017.

[6] Gradio team.

[7] I-Hung Wang, Mahardi, Kuang-Chyi Lee, and Shinn-Liang Chang. Predicting the breed of dogs and cats with fine-tuned keras applications. *INTELLIGENT AUTOMATION AND SOFT COMPUTING*, 30(3):995–1005, 2021.

[8] S. Weidman. *Deep Learning from Scratch: Building with Python from First Principles*. O'Reilly Media, 2019.