Transfer Learning with Cats Vs Dogs Dataset

Mohit Vijay Agarwal
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110065664

Prem Shanker Mohan
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110036738

Dhara Kishorbhai Hirpara
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110071938

***Abstract*—This project focuses on the classification of the Cats and Dogs datasets from Kaggle using a convolutional neural network and transfer learning. The Cats and dogs dataset is a collection of huge dataset which has manually classified pictures of cats and dogs. The Dogs vs. Cats dataset is a standard computer vision dataset that involves classifying photos as either containing a dog or cat. We study the effects of different optimizers (Stochastic Gradient, Adam and RMS Prop) and loss functions (Categorical Entropy, Mean Squared Error, and Sparse Categorical Cross Entropy)) have on the accuracy of our CNN model. Also, we use transfer learning using Inception V3. After analysis and choosing the best loss and optimization function, our model produced an accuracy of 95.00%, whereas, CNN model produced a much lower accuracy.**

***Index Terms*—Artificial Intelligence, Transfer Learning, Neural Networks, Image Classification, Inception V3**

I. Introduction

Computer Vision is one of the most exciting sub fields in the Artificial intelligence. Computer vision was one of the well established fields before even machine learning existed. However, with recent introduction of CNN, the field is brimming with possibilities. Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Using digital images from cameras and videos and deep learning models, machines can accurately identify and classify objects — and then react to what they "see."

In the 1950s, early computer vision research used some of the first neural networks to recognize object edges and sort simple objects into categories like circles and squares. The earliest commercial application of computer vision was in the 1970s, when optical character recognition was used to decipher typed or handwritten text. For the blind, this development was utilized to decipher printed text.

Facial recognition programs grew in popularity as the internet evolved in the 1990s, making massive sets of photographs available for study online. Machines can already recognize specific people in images and videos thanks to these expanding data sets.

In the subject of computer vision, there is a great deal of research being done, but it isn't simply research. Computer vision is critical in business, entertainment, transportation, healthcare, and everyday life, as demonstrated by real-world applications. The deluge of visual information streaming from smartphones, security systems, traffic cameras, and other visually instrumented devices is a primary driver for the expansion of these applications. This information might be extremely useful in a variety of businesses, but it is currently underutilized.

Image Classification and detection have become a common commodity in today's world. These techniques have been implemented in almost all systems ranging from self driving cars to the face recognition technology in our smartphones and our readily being integrated into the medical field. In the recent years, deep learning has become an ideal solution to many of the complex problems such as face recognition [1], object classification [2], gesture recognition [3], et al.

Image processing, image analysis, and machine vision are the topics of computer vision that are most closely connected. The spectrum of techniques and applications covered by these two groups overlaps significantly. This means that the fundamental methodologies utilized and developed in these subjects are comparable, implying that there is just one field with several names. On the other hand, it appears that research groups, scientific journals, conferences, and businesses must portray or advertise themselves as belonging to one of these disciplines, and so many characterizations that separate one area from the others have been provided.

In today's environment, classification and detection have become commonplace. These approaches have been integrated into practically all systems, from self-driving vehicles to facial recognition technology on our smartphones, and are now being used in the medical area as well. Deep learning has recently emerged as an appropriate solution to a number of complicated issues, including face recognition, object classification, gesture recognition, and so on.

In image recognition, the convolutional neural network (CNN) [4] is a game-changer. Hinton and his student Alex Krizhevsky designed the AlexNet network, which won the 2012 ImageNet Large Scale Visual Recognition Challenge with a performance advantage of 10% over second place in top-5 error rate, ushering in the reign of convolutional neural networks and thus the subsequent VGGNet [5], GoogleNet [6] networks.

For our research we have worked on two different methodologies one is the transfer learning using Inception V3 and other is the basic CNN model implementation. For the Inception V3 we first use the pretrained model by importing it from keras and add a few layers like the dropout and dense to it.

We run this model on the dataset that we have obtained from kaggle using Imagenet for pretrained weights. This reduces the training time and also provides a good accuracy. Lastly, we complie the model with loss and optimization functions. The project's workflow for the CNN model is as follows. We begin by extracting the features and associated class zip files from the dataset. The data must next be processed and normalized. Then we modify the image to make it easier for the CNN model to take in the input data and build a model around it. The ImageDataGenerator method is then used to generate extra data that will aid in the improvement of our model's accuracy. Finally, we'll train the model with loss functions and optimization functions. It is to be noted that the accuracy we achieve from the transfer learning methodology is sharply greater than what is obtained using the basic CNN model.

II. Problem Statement

A. Problem Description

Image recognition was one of the topics we wanted to investigate. This project details our efforts to solve the "Dogs vs. Cats" Kaggle challenge from 2013. The purpose of the original "Dogs vs. Cats" competition was to create an algorithm that could determine if photographs included a dog or a cat. It's worth noting that there was no TensorFlow or any framework like PyTorch to assist in 2013.

Our main goal of this project is to correctly classify the images using convolutional neural networks and transfer learning by adjusting the hyper parameters and finding the model that gives the highest accuracy.

B. Motivation

Along with a massive volume of visual data (about 3 billion photographs are posted on the internet every day), the computational power needed to evaluate it is now available. The accuracy rates for object recognition have increased as the area of computer vision has progressed with new hardware and algorithms. Today's systems have improved from 50% accuracy to 99 percent in less than a decade, making them more accurate than humans in swiftly reacting to visual inputs.

Cats Vs Dogs serves as a bench mark dataset to experiment cutting edge computer vision solutions and compare and contrast our performance with the performance with other algorithms.

C. Justification

Recognition of cats vs. dogs might be a crucial step toward artificial intelligence and the computer vision sector. The issue of manually written digit recognition as a model is used by the pattern recognition and machine learning groups to evaluate the categorization of models using the Cats vs Dogs dataset.

III. Literature Review

"Image Classification with Artificial Intelligence: Cats vs Dogs" [7] sheds light on the utilization of Artificial Intelligence in Image Classification. For the project demonstrated in his research Youngjun Lee, compares the Convolutional Neural network (CNN) models and Support Vector Machine (SVM) over an open source real-world dataset for image classification between dogs and cats. The data set consists of a total 25000 images divided equally for a better balance. Image classification with SVM was done in two steps, feature extraction - by converting input image to HSV space and, model classification based on the standard SVM. The CNN for this project consisted of 4 convolutional layers and 4 pooling layers with different kernel sizes. From the results obtained it can be noted that CNN performs far better than SVM and as for the choice of optimizers, Adam optimizer and LeakyReLU activation function topped the other optimizers and activation functions respectively, thereby getting an accuracy of about 92 percent.

"Multiple Classification of Flower Images Using Transfer Learning" [8] focuses on the use of pretrained models. Models such as Alexnet, Googlenet, VGG16, DenseNet and ResNet have been used following the methodology of transfer learning. The models studied are based on the Convolutional Neural Networks, executed using NVIDIA Geforce Gtx 950M GPU over the Five-Class Kaggle flowers dataset. The used dataset has approximately 800 photos each for 5 different classes of flowers. Study shows that using the transfer learning approach the amount of time and data required for training can be reduced significantly. The models proposed have been operated with a maximum of 1812 iterations. Results shown display the highest accuracy for the VGG16 model at 93.52 percent which takes comparatively larger amount of working time compared to the other less accurate but acceptable models.

"Fresh Tea Leaves Classification Using Inception-V3" [9] studies the effect of transfer learning and data augmentation using Inception V3 for the classification of fresh tea leaves. The paper discusses the various iterations of the Inception models and how they have evolved over time with an increase in the computational power by several folds. For the proposed project, a custom (self created) dataset of fresh tea leaves was used which contianed about 4000 images of 3 different classes. Data augmentation was performed over the created dataset, expanding the data augmented dataset to about 12000 images. The datasets were divided into sample sets of ratio 4:1, where both the original dataset and the data augmented dataset gave an training accuracy of 100% and validation accuracy of 95% and 98% respectively. The paper demonstrates this research to improve the production efficiency of the tea leaves and implementation of Artificial Intelligence in such industries.

IV. Components

A. Dataset

Since we are conducting research over a kaggle competition in this paper, we applied our model to the Kaggle's Dog vs Cat dataset. The database obtained has 25,000 images of different sizes equally divided among the two classes (12,500). The dataset is divided into train and test zips, which are extracted. We train our algorithm on train.zip and predict the labels for

test1.zip (1 = dog, 0 = cat). The dataset has been collected from kaggle and may not accurately represent all images, i.e., there may be a blur photo of the cat or dog in different backgrounds, or there maybe someone holding the animal.

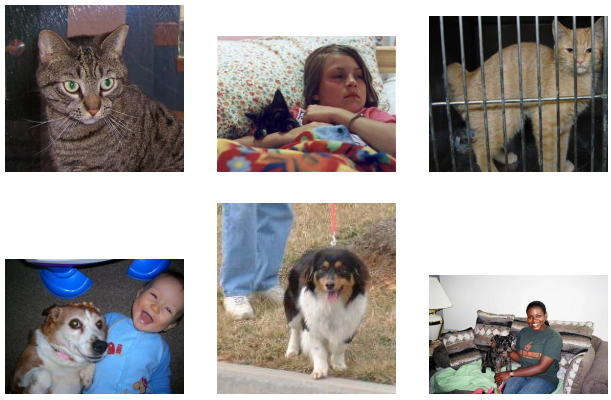

Fig. 1: Sample Images from the Dataset

B. Basic CNN Model

Convolutional Neural Networks evaluate information in the same way that a human brain does, which is why they are the best at image classification. When the network comes across a picture, it breaks it down into small chunks and examines each one separately (which is called extraction). From these sub parts, CNN pulls elements that it learned during the training phase.

A CNN typically has several layers, each of which may contain a large number of neurons. The input layer, the hidden layer, which will include the convolution and pooling layers, and the output layer, which will include the dense layer, are the three sections in general.

The convolution layer is applied to all of the training images during the training process, resulting in varied pixel values. These values are compared to the pixels in the original image, and the weights are changed to make our output look more like the original.

The convolution layer's main job is to extract information from a picture by altering and updating weights. The convolution layers extract features based on the precise placements of the feature in the image, which is a crucial consideration when using this feature extraction. So, if the position of the feature changes even little, the convolution layer will generate a different image.

The use of a pooling layer is the solution to this problem. After the convolution layer, this layer is combined. The pooling layer, like the convolution layer, applies a filter to the original picture, which is typically 2x2 pixels with a 2 pixel stride. Depending on the size of the filter used by the developer, a pooling layer will always lower the size of the image. There are two types of pooling layers that are commonly used:

- **Average Pooling**: It computes the average of all the pixel values that the filter covers.
- **Max Pooling**: The maximum pixel value from all the pixels covered by the filter is returned.

From figure 2 we can see that the proposed CNN model has five Convolutional layers, along with five max pooling layers and a flatten and a dense layer for the hidden layer and a single dense layer with activation function as *softmax* for the output layer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 128)         147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 540,929
Trainable params: 540,929
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2: Basic CNN Model Summary

C. Transfer Learning

Transfer learning is a machine learning technique in which a previously trained model is meant to be reinterpreted in a new task [10]. Various studies have indicated that using transfer learning significantly reduces the training time and cost.

1) **Better initial model:** In other methods of learning, you must create a model from scratch. Transfer learning is a better starting point since it allows people to accomplish tasks at a higher level without having to be trained.
2) **Higher learning rate:** Because the problem has already been taught for a similar task, transfer learning allows for a faster learning rate during training.
3) **Higher accuracy after training:** Transfer learning allows a machine learning model to converge at a higher

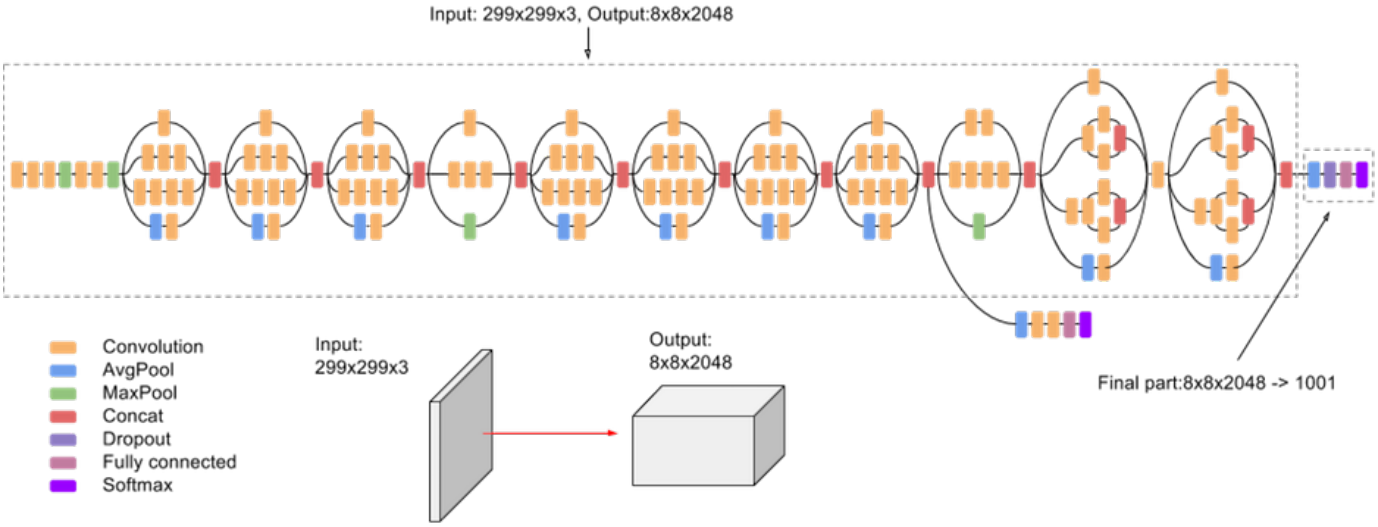

Fig. 3: Inception V3 Architecture

performance level, resulting in more accurate output, thanks to a better starting point and higher learning rate.

4) **Faster training:** Because it uses a pre-trained model, the learning can attain the target performance faster than traditional learning approaches.

Transfer learning occurs when we apply domain knowledge from one problem to another. For example, the skills learned from recognising vehicles might be applied to recognising trucks. We used a Neural Network trained using ImageNet weights, which we implemented using Tensorflow's Inception V3.

D. Inception V3

The Inception V3 is a deep learning model for image classification that uses Convolutional Neural Networks. The Inception V3 is a more advanced version of the fundamental model Inception V1, which was first released in 2014 as GoogLeNet.

Google's Inception V3 [9] includes factorization, which is considered the most significant advance over the previous version, Inception V2. Its 7x7 convolution is decomposed into two one-dimensional convolutions (1x7, 7x1), and the same is true for its 3x3 convolution (1x3, 3x1). Also, the RMSProp optimizer has been included. In this approach, the calculation can be sped up, and the convolution can be split into two, increasing the network depth and non-linearity (ReLU [11] is be carried out for each additional layer).

The ImageNet dataset was used to train the Inception V3 [12] model, which provides information that can identify 1000 ImageNet classes. The top-5 mistake rate is 3.5 percent, whereas the top-1 error rate is 17.3 percent.

We have also tweaked numerous hyperparameters to fine-tune the model in order to get the best level of accuracy. The following are the hyperparameters:

- **Loss function**: During the training process, this function assists our model in assessing the mistake and optimising the model. "Binary Crossentropy Loss Function" was the loss function employed in this research.
- **Number of classes & epoch**: We used 5 epochs to get the desired result in our project, which consists of two classes (dog and cat).
- **Optimizer**: It's the function you utilise to update the neural network's weight. We use the RMSProp optimizer in our project.

Figure 3 represents the basic architecture of Inception V3. For this pretrained model, we have specified an input shape of (150, 150, 3) instead of the default (299, 299, 3) so as to accommodate our dataset and the pretrained weights, changing the include_top to false we create a network that doesn't include the feature extraction layers, inplace of that we add the 'mixed9_1' layer to get a larger feature map.

E. Weight Initialization

The goal of this strategy is to prevent layer activation outputs in our deep neural network from exploding or disappearing during the forward pass. When there are a lot of layers, some weights can get exponentially huge, to the point where the computer becomes incapable of recognising them. As a result, the weight may explode. On the other hand, if the weight is small, it is possible that they will shrink to the point where the computer will perceive them as zero. Thus, we use weight initialization to avoid this problem.

For this research we have used Imagenet for pretrained weights stored as .h5 file obtained from the google api storage. ImageNet is a database of images arranged according to the WordNet hierarchy, with hundreds of thousands of photos depicting each node of the hierarchy. Many advances in computer vision research have been fueled by network architectures

measured against ImageNet, including transferring to new datasets [13] [14], object detection [15], image segmentation [16] [17], and perceptual metrics of images [18]. Researchers can use the data for non-commercial purposes for free.

Fig. 4: Sample of ImageNet dataset

F. Optimizers

Optimizers are algorithms that help minimize the losses by letting us know how to change weights and what is the learning rate of the neural network [19]. The optimizers we have used are mentioned below:

- **Adam Optimizer**: Adam(Adaptive Moment Estimation) is one of the most popular gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. Adam is described as a combination of the AdaGrad and RMSProp algorithms and can handle sparse gradients on noisy problems [19].
- **SGD Optimizer**: SGD (Stochastic Gradient Descent) updates the model parameters one by one and is a variation of the Gradient Descent algorithm. The Stochastic Gradient Descent requires less memory and allows the updated dataset of large datasets. However, it does not always guarantee good convergence [19].
- **RMSProp Optimizer**: RMSProp (Root Mean Square Propagation) is quite similar to SGD and is considered a special variation of AdaGrad. Here, the learning rate is an exponential average of the gradients instead of the cumulative sum of squared gradients [19].

G. Evaluation metrics

Accuracy: The ratio of correct forecasts to total predictions is known as accuracy. When it comes to measuring the algorithm's performance in a transparent manner, the accuracy metric is used. It is usually estimated as a percentage after the model parameters have been determined. It's a metric for determining how close the model's forecast is to the actual data. The accuracy formula can be seen below.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

Loss: When trying to improve an algorithm, a loss function is used. The loss is determined by the model's training and validation, which defines how well the model weights updates to provide the most accurate model. The loss value represents how well or poorly the model behaves after each optimization step. The following are the various loss functions used in this project:

- **Mean Squared Error**: This loss function is used for regression. Here, the loss is calculated by taking the mean of squared differences between the target and predicted values. [20]
- **Squared Hinge Loss**: The square of the score hinge loss is calculated in squared hinge loss. When there is a mismatch in the sign between the actual and predicted class values, the hinge loss function assigns greater error, encouraging examples to have the proper sign. Performance with the hinge loss has been varied, with some reports claiming that it outperforms cross-entropy on binary classification problems. [20]
- **Binary Crossentropy**: It's designed for binary classification with target values in the range of 0 to 1. For predicting class 1, cross-entropy will compute a score that summarises the average difference between the actual and predicted probability distributions. A perfect cross-entropy value is 0 when the score is minimised. [20]

As the final part of our evaluation, the model was examined against the test set. The data required for this was created, and the model's 'evaluate' method in TensorFlow was used to compute the metric values for the trained model. This method returns the loss value and metrics values for the model in test mode.

V. Implementation Details

In this project, for the basic CNN model for classification of dogs vs cats we first need to preprocess the images. We accomplish this by extracting the content of the .zip files into two different folders: train and test.

```
#Unzips files
with zipfile.ZipFile(os.path.join(root,'train.zip'), 'r') as zip_ref:
    zip_ref.extractall(os.path.join(root_temp))
with zipfile.ZipFile(os.path.join(root,'test1.zip'), 'r') as zip_ref:
    zip_ref.extractall(os.path.join(root_temp))
```

Fig. 5: Unzipping the zip files

We then confirm the unzipped dataset present in our file system. The dataset should contain 25000 training images and

12500 testing images. Then, we define a pandas dataframe adding a binary and a categorical label for each image.

```
train_images = [fname
                for fname in os.listdir(os.path.join(root_temp,'train'))]
test_images = [fname
               for fname in os.listdir(os.path.join(root_temp,'test1'))]

print('total number in training set ', len(train_images))
print('total number in test set ', len(test_images))
```

```
total number in training set  25000
total number in test set  12500
```

Fig. 6: Validation of Dataset

The train dataframe contains each photo filename, category, and binary labels, and it is this data that we will use to train our CNN, whilst the test dataframe has no labels and may only be used to submit our predictions to the competition. To appropriately test our model performance, we will partition our labeled data into a train set and a validation set in subsequent cells of the project.

After that, we'll normalize and restructure the data. We do both of these things because when we train our CNN network, we'll be multiplying (weights) and adding to (biases) these initial inputs in order to generate activations, which we'll then backpropogate using gradients to train the model. We want each feature in this procedure to have a comparable range (0-1) so that our gradients don't get out of hand (and that we only need one global learning rate multiplier). We also on encode out label data to 0 and 1 since there are only 2 classes.

Let's have a look at our model's setup now. The Keras Sequential API was utilized, which allows you to add one layer at a time. We begin with the input and then go on to our first convolutional layer (Conv2D). We picked one 32-filter, 64-filters for the second and third levels, 128-filters for the fourth layer, and 512-filters for the final Conv2D layer. Our CNN model is built to locate and isolate all of the valuable characteristics in photos that have been transformed.

The pooling (MaxPool2D) layer is the next important layer in the CNN. This layer serves as a downsampling filter for our photos. It looks at the two pixels next to it and picks the highest value. These layers are primarily used to minimize computing costs, as well as to reduce overfitting to some extent. Our CNN can integrate local characteristics and grasp more about the global properties of our MNIST picture by combining convolutional and pooling layers. The rectifier (activation function max(0,x)) is called'relu.' This rectifier activation function is utilized to give the network non-linearity, which helps the model fit better.

Flatten converts the features from final feature maps into a single 1D vector. After some convolutional/max pool layers, you may employ fully connected layers, hence this flattening phase is necessary. It incorporates all of the previously discovered local characteristics from the convolutional layers. Finally, we employed Artificial Neural Networks (ANN) classifiers in two fully connected (Dense) layers. We utilize a softmax function in the last layer (Dense(10,activation="softmax") to net the output distribution of probability of each class. To

```
[ ] pre_trainned_model = tf.keras.applications.InceptionV3(include_top = False,
                                                           weights = 'imagenet',
                                                           input_shape = (150,150,3))

for layer in pre_trainned_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/in
87916544/87910968 [==============================] - 1s 0us/step
87924736/87910968 [==============================] - 1s 0us/step
```

```
[ ] last_layer = pre_trainned_model.get_layer('mixed9_1')

pretrainned_last_output = last_layer.output

x = tf.keras.layers.Flatten()(pretrainned_last_output)
x = tf.keras.layers.Dense(512, activation = 'relu')(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(1, activation = 'sigmoid')(x)

tf_model = tf.keras.models.Model(pre_trainned_model.input , x)
```

Fig. 7: Training on Inception V3

improve the performance of this model, we employ transfer learning. What this implies this, we are taking a pretrained model of inception V3 and we are adding more CNN layers and training that with our cats vs dogs dataset as showing in figure 7. The idea behind this act is that the pretrained weights would help us classify the dataset faster and more accurately.

VI. Results and Analysis

The basic CNN model employed for this project gives an accuracy of approximately 84%, with a loss of about 14% for both training and validation. For obtaining this result we have employed the "Mean Squared Error" Loss function along with the RMSProp optimizer.

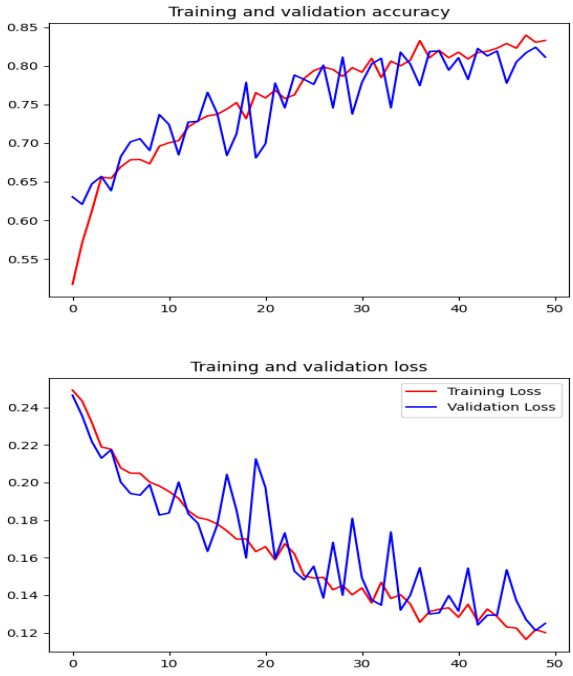

Fig. 8: Accuracy and Loss Graphs for CNN

Various loss functions were implemented on the CNN model, and the binary crossentropy outperformed all the other losses to provide an accuracy of approximately 70% when ran only for 5 epochs, which will increase significantly on increasing the number of epochs. Figure 9 shows a comparison between all the loss functions ran over the CNN model.

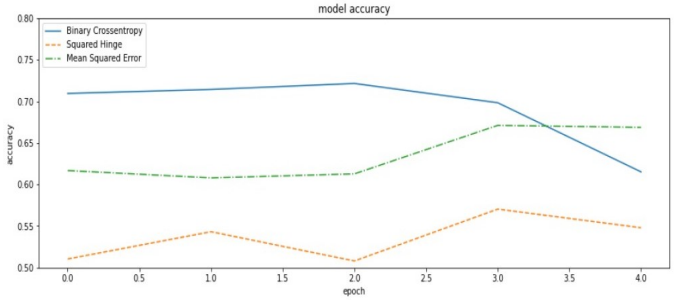

Fig. 9: Comparing Losses

Comparing Inception V3 with the basic CNN model, there is a tremendous boost in the accuracy and a significant drop in the loss. For this model, we get an accuracy of about 93% and a loss of about 15% for both the training and validation, which is a great improvement over the metrics of the basic CNN model. For the Inception V3 we used Binary Crossentropy for the loss estimation and RMSProp for our optimizer, which gave us the best results compared to other loss functions and optimizers when working with less number of epochs.

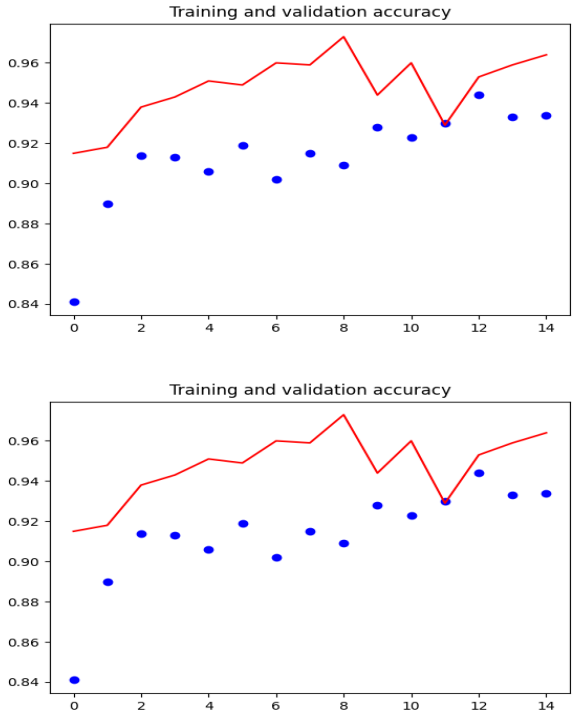

Fig. 10: Accuracy and Loss Graphs for Inception V3

VII. Conclusion

A. Summary

In this research, we discovered how to develop a Convolutional neural layer for Cats vs Dogs dataset and optimization from scratch and correlated transfer learning models to get the best result. Specifically, we learned the things defined below:

1) Develop a test model, robust evaluation of a model, and establish a baseline performance for a classification task.
2) Explore the transfer learning models to not only improve the performance but also decrease the training time.
3) Develop a finalized model, evaluate the performance, and use it to get an accurate result of images.

From the project we can conclude that the best loss function and optimization function was categorical crossentropy loss and adam respectively having been run on a large number of epochs over the Inception V3 transfer learning model. This resulted in our model having an accuracy of approximately 93.00% in classifying the Cats vs Dogs dataset.

For the future iterations of this model, we plan to compare more types of transfer learning models with custom CNN models to get an even better accuracy. This will help us in exploring more pathways into this field.

Acknowledgement

We're privileged that Dr. Robin Gras gave us a chance to perform research on a kaggle competition in this tough time and taught us various related topics to this. Without his guidance and proper knowledge about research, it would be more difficult to complete the study.

The project code and final results are accessible on Google Colab: shorturl.at/eqzKR

References

[1] Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997) "Face recognition: A convolutional neural-network approach."IEEE transactions on neural networks, 8(1):98-113.

[2] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ...& Rabinovich, A. (2015) "Going deeper with convolutions." in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[3] Bobić, V., Tadić, P., & Kvaščev, G. (2016, November) "Hand gesture recognition using neural network based techniques." in Neural Networks and Applications (NEUREL), 2016 13th Symposium on (pp. 1-4). IEEE.

[4] LeCun Y, Boser B, Denker J S, Henderson D, "Backpropagation Applied to Handwritten Zip Code Recognition," Neural computation, vol.4, no.1, pp.541-551, 1989.

[5] K. Simonyan, A. Zisserman. "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going deeper with convolutions," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1-9, 2015.

[7] Youngjun Lee, "Image Classification with Artificial Intelligence: Cats vs Dogs", (2021) 2nd International Conference on Computing and Data Science (CDS), (pp. 437-441).

[8] E. Cengıl and A. Çinar, "Multiple Classification of Flower Images Using Transfer Learning", 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), 2019, (pp. 1-6).

[9] Y. Qian et al., "Fresh Tea Leaves Classification Using Inception-V3," 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), 2019, pp. 415-419, doi: 10.1109/ICICSP48821.2019.8958529.

[10] Guo Z, Chen Q, Wu G, Xu Y, Shibasaki R, Shao X., "Village Building Identification Based on Ensemble Convolutional Neural Networks". Sensors.;17(11):2487, 2017.
[11] A. Krizhevsky, I. Sutskever, G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks," Neural Information Processing Systems, pp.1097–1105, 2012.
[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, John Shlens, Zbigniew Wojina, "Rethinking the Inception Architecture for Computer Vision," IEEE Conference on Computer Vision and Pattern Recognition, pp.2818-2826, 2016.
[13] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In International Conference on Machine Learning, pages 647–655, 2014.
[14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 512–519. IEEE, 2014.
[15] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In IEEE International Conference on Computer Vision (ICCV), pages 2980–2988. IEEE, 2017.
[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4):834–848, 2018.
[18] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (ECCV), pages 694–711. Springer, 2016.
[19] Musstafa. "Optimizers in Deep Learning", Available from: https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0
[20] Jason Brownlee "How to Choose Loss Functions When Training Deep Learning Neural Networks", Deep Learning Performance, https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/, 2019