Capstone Project

Machine Learning Engineer Nanodegree

Mohammed Ehsan Ur Rahman

28th July 2021

Major Project Development Stages.

1. Definition

Project Overview

This is a Udacity specific project and came with pre-stated goals and a Jupyter notebook to complete. The project is to use Convolutional Neural Networks to Identify Dog Breeds
Dog breed classification is really challenging due to the minimal inter-class variations. A Convolutional Neural Network (CNN) model can be used for this task. Due to the existence of a huge variety of species and organisms, research on technology to classify them is crucial. Hsu, David from Stanford University published a paper about using convolutional neural networks (CNN) to classify dog breeds. The final project will be able to identify the breed from 133 given breeds given an image of a dog or if given an image of a human it will identify the breed which is similar to the human.
A computer vision project which is the subset of deep learning techniques used and applied to solve traditional image processing tasks. The most used algorithms and architecture are neural networks and backpropagation. Recognising the breed of dogs is a challenging task in image classification sub branch of computer vision. There are hundreds of breeds in existence which are grouped into 10 distinct groups according to physical characteristics. Classification is a complex task as there are lots of classes and just subtle differences between each class. It is also computationally and memory wise expensive due to the number of features involved. Remembering all the breeds and distinguishing between similar breeds cannot be easily done by even humans. The current task involves two subparts one is image recognition, and the other is image classification which uses machine learning techniques like convolutional neural network.

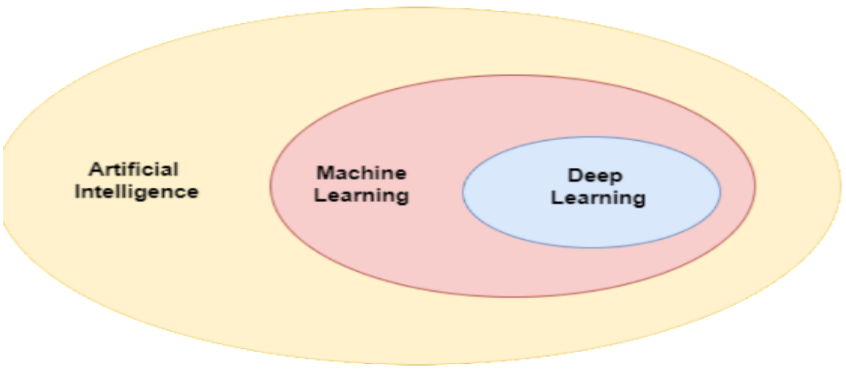


Figure 1Domain of Deep Learning is a subset of Machine Learning which is a subset of Artificial Intelligence

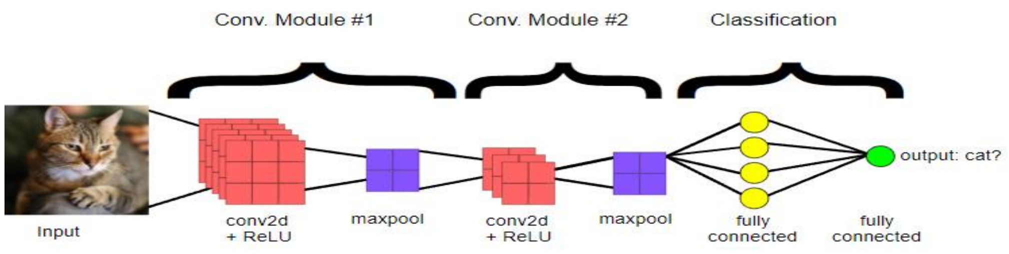


Figure 2 A typical model of CNN

Problem Statement — a problem being investigated for which a solution will be defined
The aim of the project is to build a pipeline to process real-world, user-supplied images. The algorithm will identify an estimate of the dog's breed given an image. When the image is of a human, the algorithm will choose an estimate of a dog breed that resembles the human. If neither a dog or a human is detected,
then an error message is output. Therefore, the models in place should be capable of detecting a dog or human in an image, classify the dog to its breed and classify a dog breed that the human resembles.

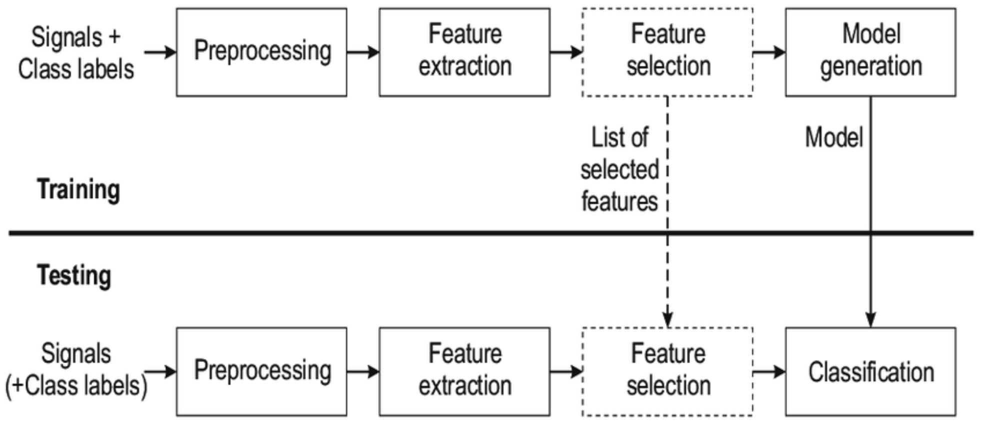


Figure 3 Workflow of training and testing

Given an image of a dog, the model will identify an estimate of the canine's breed. If supplied an image of a human, it will identify the resembling dog breed. Various stages of the project are:

1. Pre-processing and import Datasets
2. Feature extraction: Detect Humans using OpenCV's Haar feature-based cascade classifiers.
3. Model building and classification: Detect Dogs using a pre-trained VGG-16 model.
4. Create a CNN to Classify Dog Breeds (from Scratch)
5. Create a CNN to Classify Dog Breeds (using Transfer Learning)
6. Write an Algorithm to print the breed if a dog is detected or the breed that resembles the human if a human is detected or an error message if none of them is detected.
7. Testing the Algorithm on some images from the datasets

Metrics

Accuracy will be used to evaluate both the solution model and benchmark model. Accuracy will
be calculated with the help of the confusion matrix.
$Accuracy = (TP + TN) / (TP + TN + FP + FN)$
TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative
True Positive(TP)= an outcome where the model correctly predicts the positive class
True Negative(TN)= an outcome where the model correctly predicts the negative class
False Positive(FP)= an outcome where the model incorrectly predicts the positive class &
False Negative(FN)= an outcome where the model incorrectly predicts the negative class.

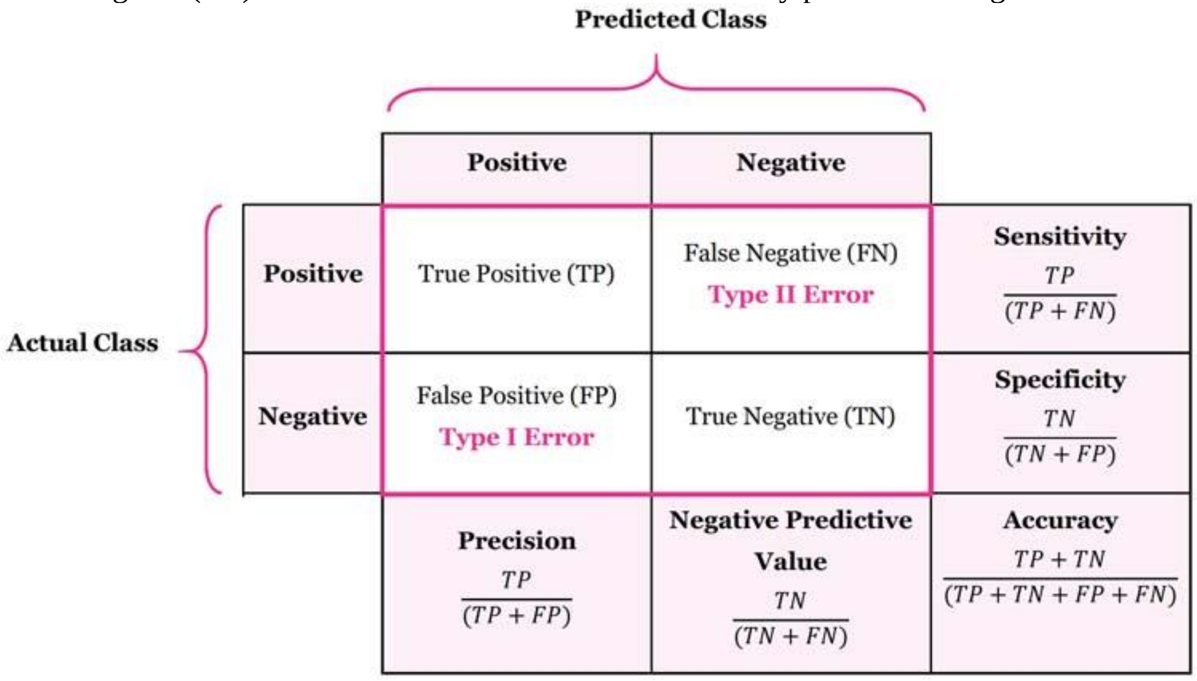


Figure 4 Confusion matrix

The accuracy requirements of the project are as follows:
1. The CNN built from scratch was required to have an accuracy greater than 10%.
2. The CNN built using transfer learning was required to have an accuracy of 60% at least.

2. Analysis

Data Exploration

Datasets and Inputs — data or inputs being used for the problem
There is the dog dataset containing 133 breeds and each breed contains 8 images. There is also the human dataset[3] which contains images, and labels(first names and last names). There is a total of 13233 human images and 8351 dog images. The human dataset will be used to detect human faces in images using OpenCV's implementation of Haar feature-based cascade classifiers. The dog dataset is used to detect dogs in images using a pre-trained VGG-16 model. The datasets of human and dog images is justified for this project given the problem statement above.
The dogs' datasets are downloaded from the link dog dataset and placed it in the repo, at location path/dog-project/dogImages. The dogImages/ folder will be containing 133 folders

each having images of respective breed's dogs, and each folder corresponding to a different dog breed.
The humans' datasets are downloaded the human dataset and placed it in the repo, at location path/dog-project/lfw.
Example images:

Figure 5 Dog of breed: Affenpinscher

Figure 6 Human Face

Algorithms and Techniques

CNNs are a kind of deep learning architectures that can learn to do things like image classification and object recognition among many others. They keep track of spatial information and learn to extract features like the edges of objects in something called a convolutional layer. The convolutional layer is produced by applying a series of many different image filters called convolutional kernels to the input image.
This project requires two CNNs (model from scratch and model from transfer learning) mainly for classifying the breed and one for detecting dogs.
For detecting dogs, a pre-trained VGG-16 model was used, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).
The model from scratch was implemented with 3 convolution layers for extracting the features and 2 fully connected layers to act as the classifier.
Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task. For transfer learning, I picked the pre-trained Resnet-50 model and froze its layers and replayed its fully connected layer with one that suits our purpose and trained the model to update the weights of the replaced layer.

Benchmark

The implementation of the model from scratch had a Test Loss of 4.051445 and Test Accurac y: 13% (113/836). My implementation of the model from transfer learning had a test loss of 1 .3424 and test accuracy of 87%.

3. Methodology

Data Pre-processing

The images were resized and cropped to 224x224 pixels to achieve an input tensor of shape 224x224 which suits the pre-trained model and the model from scratch.
Images for the training set had undergone data augmentation to prevent overfitting and to get more generalized training data. Even the validation and testing data was exempted from augmentation, but they were resized and cropped to 224x224 like the training set.
After resizing and cropping, the image data were converted to tensors, and they were normalized for faster computation and to suit the data types processed by Pytorch.

Implementation

The implementation of this project involved 3 major parts: the CNN from scratch, the CNN from transfer learning and the final app.
The solution proposed for the problem given:
1. Firstly a Haar cascade based built-in face detector is used in the project that is an OpenCV's implementation of Haar feature-based cascade classifier. It is used to detect human faces from the given image.
2. A pre-trained VGG-16 based model with trained weights on ImageNet, will be used to detect dogs. The pre-trained model should be checked that it returns indexes from 151 to 268 inclusive, that is, it must include categories from 'Chihuahua' to 'Mexican Hairless'.
3. A CNN model is then built from scratch to classify dog
breeds, that is, transfer learning cannot be used just yet. This model should surpass a test accuracy of 10% set by Udacity because the model is being built from scratch so classifying similar breeds can be a challenge however transfer learning will greatly improve this.
4. Finally, a transfer learning will be used with a ResNet50 model to significantly boost the accuracy of the CNN model. It should surpass the 60% test accuracy set by Udacity.

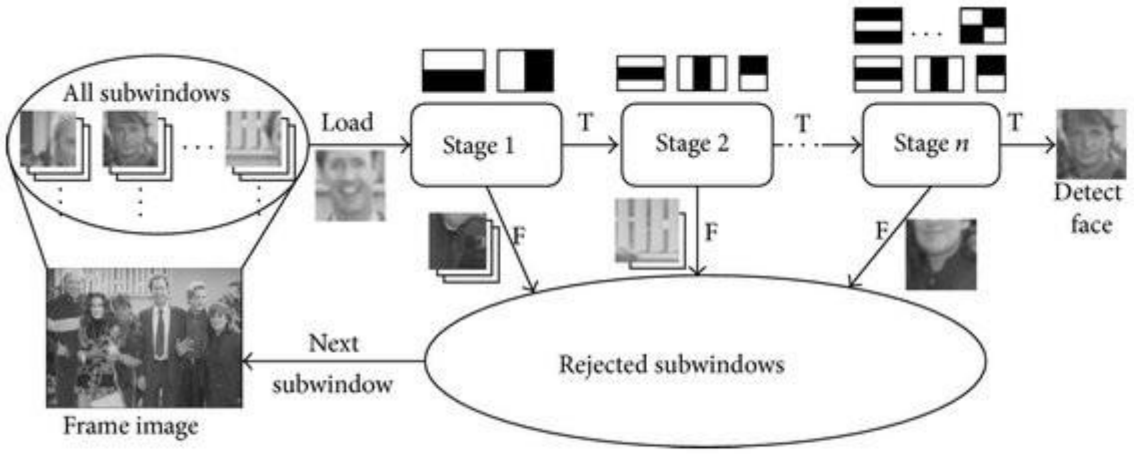


Figure 7 Haar Cascade Detector

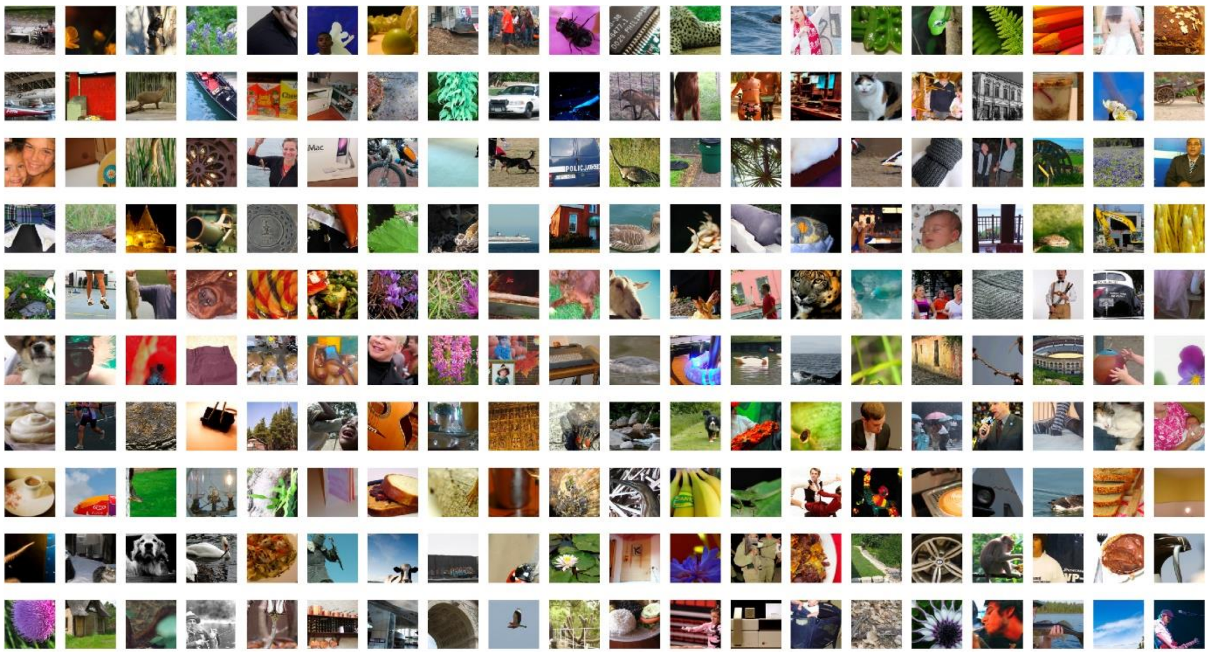


Figure 8 Examples from ImageNet

The code snippet below shows the architecture of the CNN from scratch.

```
class Net(nn.Module):
    ### TODO: choose an architecture, and complete the class
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 16, 4, stride = 2, padding = 1)
        self.conv2 = nn.Conv2d(16, 32, 4, stride = 2, padding = 1)
        self.conv3 = nn.Conv2d(32, 64, 4, stride = 2, padding = 1)

        #linear layers
        self.fc1 = nn.Linear(64*7*7, 784)
        self.fc2 = nn.Linear(784, 133)

        #dropout layer
        self.dropout = nn.Dropout(p = 0.3)

        self.batch_norm1 = nn.BatchNorm2d(32)
        self.batch_norm2 = nn.BatchNorm2d(64)
        #maxpool layer
        self.maxpool = nn.MaxPool2d(2, 2)

    def forward(self, x):
        ## Define forward behavior

        x = F.leaky_relu(self.conv1(x), 0.2)
        x = F.leaky_relu(self.batch_norm1(self.conv2(x)), 0.2)
        x = self.maxpool(x)
```

```
x = F.leaky_relu(self.batch_norm2(self.conv3(x)), 0.2)
x = self.maxpool(x)

x = x.view(-1, 64*7*7)

x = F.relu(self.fc1(x))
x = self.dropout(x)

# No relu here, we use CELoss
x = self.fc2(x)

return x
```

The input was a tensor of shape 3x224x224 which had undergone pre-processing. After passing through three convolutional layers, we get an output tensor which is then flattened into a vector and passed through two fully connected layers then a dropout layer, the convolutional layers are followed by batch normalization layers.
The fully connected layers and convolutional layers use a Rectified Linear Unit (ReLU) activation function.
The final fully connected layer has an output shape of 133 corresponding to the number of breeds.
The graph below depicts the ReLU activation function.

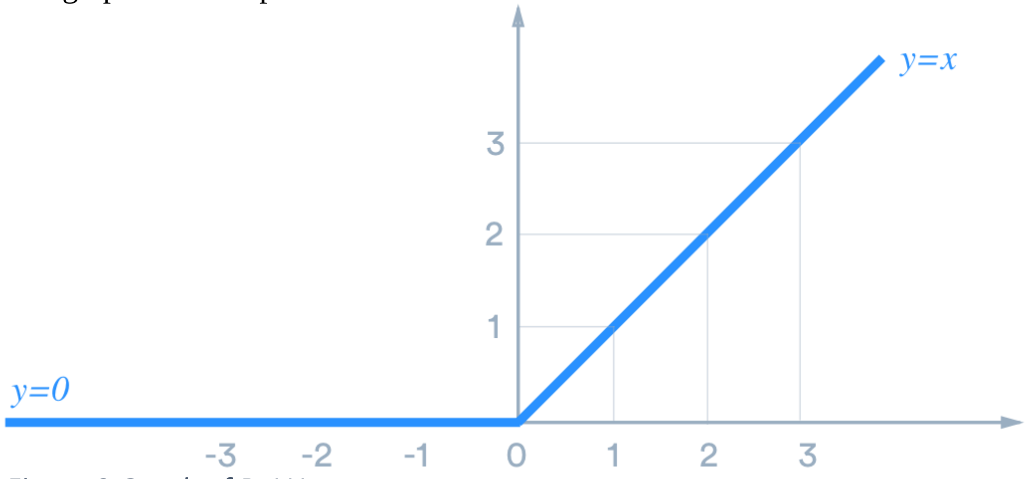


Figure 9 Graph of ReLU

For the transfer learning model, I picked a Resnet-50 model pre-trained on the ImageNet dataset.
The figure below illustrates the architecture of the Resnet-50 model.

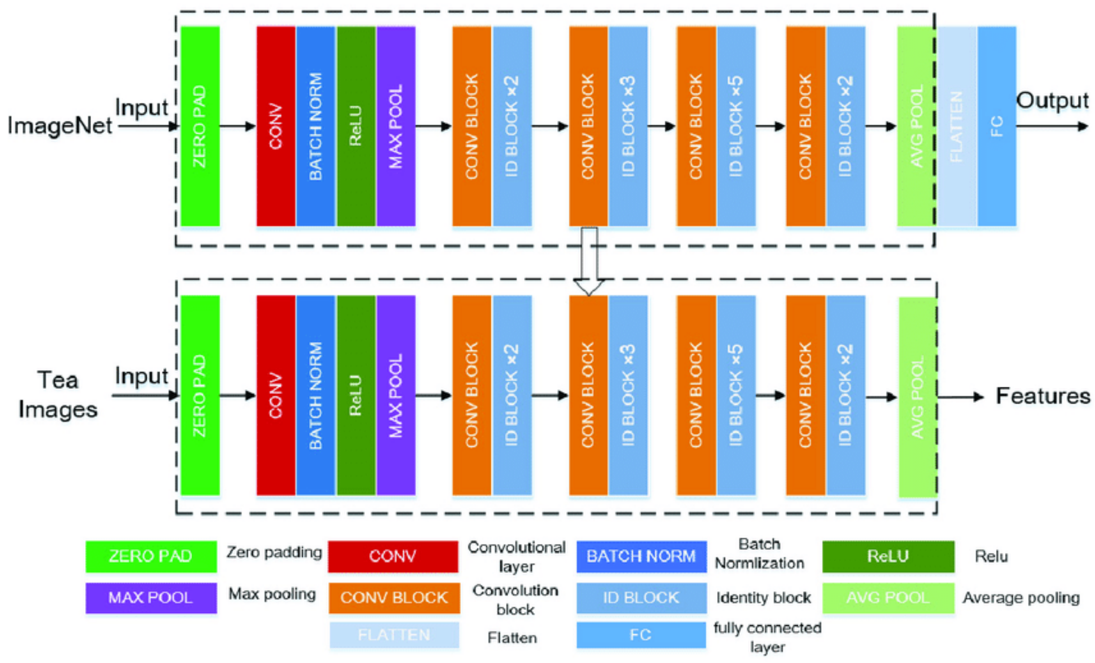


Figure 10 RESNET50 Architecture

Then, I froze all the layers in the pre-trained model and replaced its final fully connected layer with a new layer which output features equal to the number of classes in our task, i.e. number of breeds.
For both the above models, I used Cross entropy loss as criterion and Adam as the optimizer with a learning rate of 0.0001.
Finally, an algorithm was required to connect the face detector, dog detector, and CNN (transfer learning model). If a human face is detected by the face detector (OpenCV's Haar cascade classifier), the programme will provide the breed that is identical to the face. If the dog detector (pre-trained VGG-16 model) recognises a dog, the algorithm will guess the breed. The algorithm will output an error message if both detectors return false.

The code snippet below illustrates the final app.

```
def run_app(img_path):
    ## handle cases for a human face, dog, and neither
    if face_detector(img_path):
        print("It's a human!")
        predicted_breed = predict_breed_transfer(img_path)
        image = Image.open(img_path)
        plt.imshow(image)
        plt.show()
        print("And it resembles a ", predicted_breed)
        print()

    elif dog_detector(img_path):
        print("A dog is detected!")
        predicted_breed = predict_breed_transfer(img_path)
        image = Image.open(img_path)
        plt.imshow(image)
        plt.show()
        print("The dog is a ", predicted_breed)
        print()

    else:
        print("Error!")
        image = Image.open(img_path)
        plt.imshow(image)
        plt.show()
        print('\n')
```

Refinement

Both models' performance was originally subpar, and they failed to meet the benchmark standards. I included a dropout layer between the fully connected layers because the model from scratch was overfitting and overall accuracy was not adequate. This improved the accuracy significantly, and I next tested raising the dropout probability from 0.25 to 0.3, with good results. In terms of the model's architecture, I went through several iterations with various numbers of completely linked layers and varied numbers of input and output features.

For the transfer learning model, I tried variations in the new layer replacing the fully connected layer of resnet-50. Changing the optimizer from stochastic gradient descent to Adam improved the performance and I decided to go with a learning rate of 0.0001 after 0.1, 0.01 and 0.001 were showing poor performance.

4. Results

Model Evaluation and Validation

It was already recommended in the given content to use the model from transfer learning in the final app. It was an obvious choice as the model from transfer learning outperformed the model from scratch by far. The models were selected during training, during each epoch the validation accuracy of the model was compared to the one with the minimum validation accuracy. And if it was lower, the present model would be saved. Then the models were tested against previously unseen data from a testing dataset and testing accuracy was calculated. This testing accuracy was compared with the benchmark requirements. Both the models passed the benchmark tests.
The model from transfer learning performed really well and proved robust enough to solve the task of breed classification.

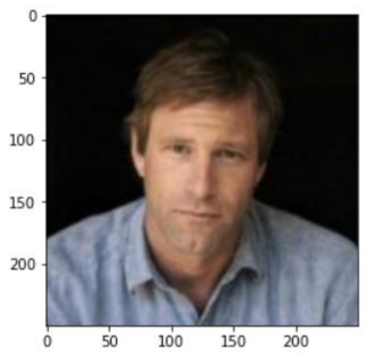
Justification

The model from transfer learning outperformed the benchmark by 27% and it seems good enough to be used in the application.
Model from scratch:
● Benchmark requirement: 10% accuracy
● Testing accuracy: 13% (107/836)
Model from transfer learning:
● Benchmark requirement: 60% accuracy

● Testing accuracy: 87% (727/836)

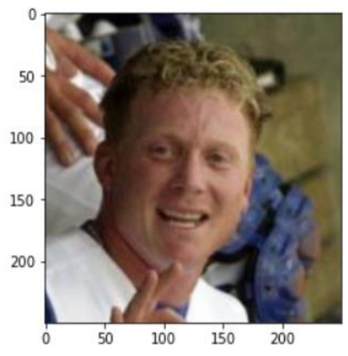
Visualizations:

```
It's a human!
```



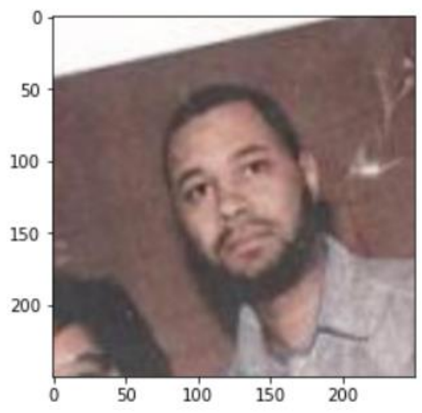
```
And it resembles a  Affenpinscher
```

```
It's a human!
```

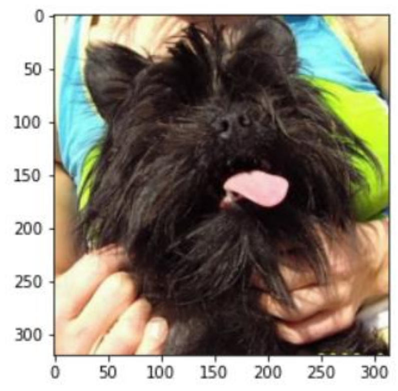


```
And it resembles a  Dogue de bordeaux
```

```
Error!
```

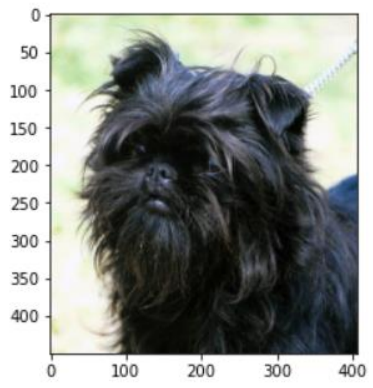


```
A dog is detected!
```



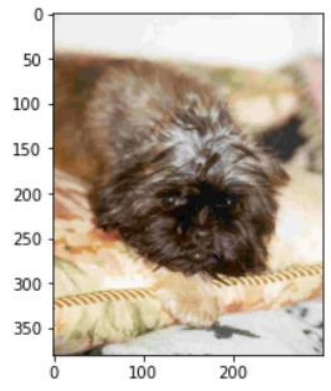
```
The dog is a  Affenpinscher
```

```
A dog is detected!
```



```
The dog is a  Affenpinscher
```

```
A dog is detected!
```



```
The dog is a  Affenpinscher
```

5. Conclusion

This project was quite a journey of learning from the initial proposal to the final report at hand. This capstone project really illustrates how to solve a real-world machine learning problem and the pipeline in solving a problem. I picked this particular project of dog breed classification (using CNNs) as I was really interested in Deep Learning, and I saw this as an opportunity to work on a real-world problem with a deep learning algorithm (i.e., convolutional neural networks) and the notion of it excited me.

I learned that most parts of the pipeline in solving a machine learning problem is expensive in terms of computing resources and time. The model took a lot of time to train even with a GPU and CUDA which highlighted the importance of making smart decisions over trial and error in terms of tuning architectures and parameters. I also learned the importance of documentation in peer-reviewed projects and journals.

Improvement

I could improve my final model using transfer learning and build it from the ground up by training it on more data that had been augmented, which would generalise the model. I could explore other architectures for a model from scratch, but this would require additional computational power, particularly more powerful GPUs. I could try different pre-trained models for the transfer learning model to see if they perform better than my Resnet-50 model. It would be ideal if the final software could be implemented as a mobile or online application that can recognise dog breeds in real time. The dog-loving community would warmly welcome it. Alternatively, we may expand the app and include models to detect birds, for example.

References:

1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.
2. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.
3. Liu, Jiongxin, et al. "Dog breed classification using part localization." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2012.
4. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
5. Cuimei, Li, et al. "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers." *2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*. IEEE, 2017.
6. https://neurohive.io/en/popular-networks/vgg16/
7. https://iq.opengenus.org/resnet50-architecture/
8. https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d#:~:text=Accuracy%20is%20an%20evaluation%20metric,True%20Positives%20and%20True%20Negatives.
9. Emami, Shervin, and Valentin Petrut Suciu. "Facial recognition using OpenCV." *Journal of Mobile, Embedded and Distributed Systems* 4.1 (2012): 38-43.

10. https://www.mygreatlearning.com/blog/resnet/
11. https://analyticsindiamag.com/why-resnets-are-a-major-breakthrough-in-image-processing/
12.
https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=ae4f3238-99a5-4811-b336-c128f300542b&bsft_uid=1c1c0b62-d63f-411e-85d4-d6f8964c8989&bsft_mid=e9b76282-2045-4474-9572-85aa21d76f07&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=9ab4e5d6-548d-465f-8a82-e69c9edd9465&bsft_mime_type=html&bsft_ek=2021-07-28T06%3A17%3A09Z&bsft_aaid=8d7e276e-4a10-41b2-8868-423fe96dd6b2&bsft_lx=2&bsft_tv=5#!/reviews/3069480
13