**Snapshot Serengeti Image Classification by CNN**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**ABSTRACT**

Convolutional Neural Network (CNN) is a type of deep learning neural networks. CNN has become a breakthrough in image recognition and the state-of-art computer vision technique. CNN works great for image classification because CNN uses convolution operations as well as pooling technique to reduce the high dimensionality of the original image data into lesser dimensional data, which then decreases the neural network’s complexity and prevents overfitting. Today CNNs can be found at the core of every bit of our lifestyle from photo tagging in social media to self-driving cars. In other words, CNNs have been proven to be fast and efficient in analytics. The objective of the project is to create transfer learning models with a pre-trained model, VGG19 and a CNN model built from scratch for a multiclass image classification. Successful product of this project will have a real-world utility, which can be used to automate the burdensome methods of manual identification of objects.

**Keywords**: CNN, transfer learning, classification

**1 Introduction**

**1.1 Snapshot Serengeti**

Snapshot Serengeti is a long-term monitoring program that is part of the Serengeti Lion Project. It consists of 255 trap cameras that are used to detect the lions in addition to other animals living on the Serengeti. The datasets include over 2,688,077 images of animals, which are shared on the Zooniverse platform. In this project, there are 5 classes to be identified: zebras, impalas, elephants, cheetahs, and lions.

**1.2 CNN and Image Classification**

Convolutional Neural Network (CNN) is a breakthrough in building models for image classification. CNNs can do many things that humans can do much faster and more efficiently. Humans recognize images by observing abstract features of an object whereas CNNs use neurons in the visual cortex that are sensitive to specific vision. Each neuron in the network looks for specific features and all of these neurons together can produce a visual perception.

There are a few layers we need to know in order to fully understand the architecture and processes of CNN: input layer, convolutional layer, pooling layer, fully connected layer, activation function (typically softmax) layer, and then output layer. First, the input layer is the layer that contains the input image, or data. Then, each of the input images passes through a series of convolutional layers so that important features of the input image are extracted. The important features are then applied by filters, pooling, fully connected layers, and softmax layer to reach the output layer. Output layer is in the form of one-hot-encoding, thus containing a class label.

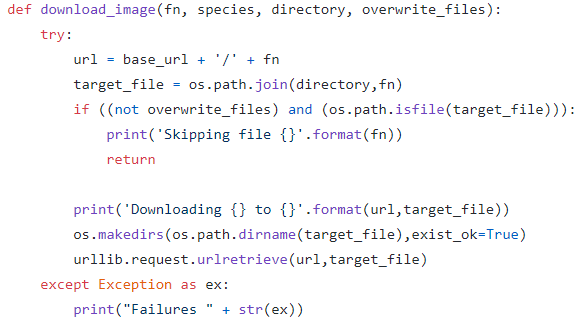
Diagram, engineering drawing

Description automatically generated

**Figure 1: General CNN architecture**

**2 Data Collection**

The datasets can be found from the website called *Labeled Information Library of Alexandria: Biology and Conservation* *(LILA BC)*. The metadata is categorized by seasons and can be downloaded in zip files. The metadata includes over 63 different species with zebra, impala, and elephant having the largest samples sizes. For each animal, the metadata can be downloaded first. Once the metadata is downloaded, a small number of images (between 100-200 images) must be individually collected. The individually collected images are chosen if they capture the animal in a close distance and only include one animal only. Once the images are first picked this way, they proceed to the next step of preprocessing and cleaning.

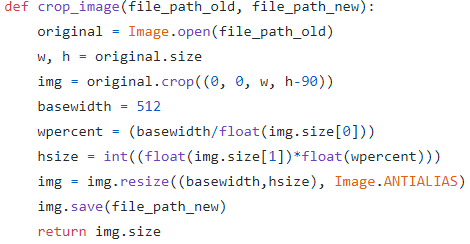


**Figure 2: download images**

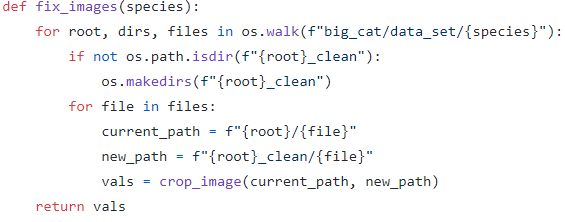
**3 Data Preprocessing & Cleaning**

Data processing is defined as a series of operations on data, especially by a computer, to retrieve, transform, or classify information. Data processing is an important process before building models on the dataset. It helps improve the data quality and in doing so, increase overall productivity. One of the most important aspects for this project is to fetch the highest quality of images possible.

Extracting images for the desired species from the metadata is the first step. Once the images are extracted, the images need to be moved to the root level of specific folders in the user’s local drive before the image cleansing process. Since the images are not in the same shape or may have the stamps at the bottom of the images, it is important to remove the noises before feeding them to the models. The crop\_images and fix\_images functions can remove noises in the images (crop) and resize the images.



**Figure 3: code to crop and remove noise images**



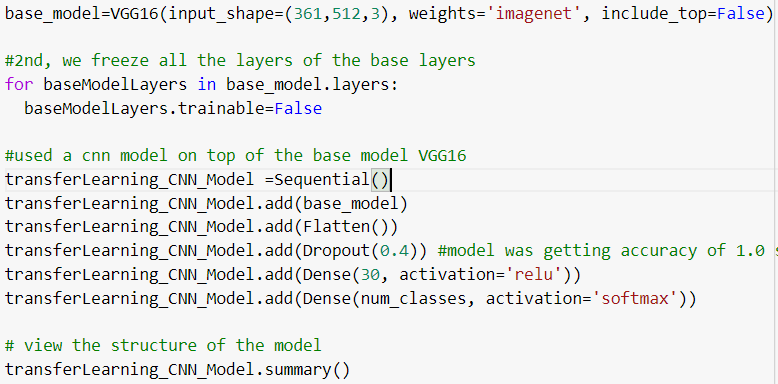
**Figure 4: code to fix images**

**4 Models Used**

One of the most famous techniques in image classification is transfer learning from a pre-trained network. A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. For this project, VGG-19, ResNet50, and a scratch CNN model are used.

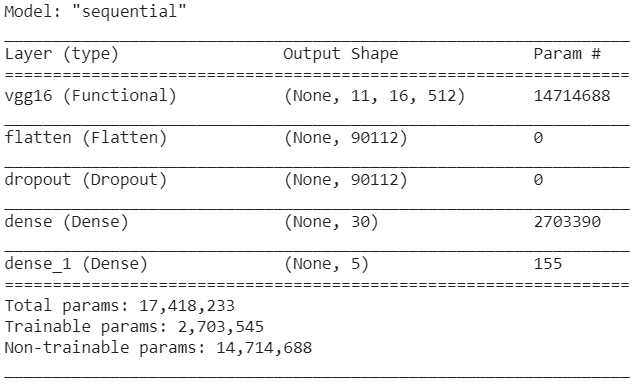
**4.1 VGG-16**

VGG-16 is a pre-trained CNN with 16 deep layers and is capable of learning rich feature representations for a wide range of images. First, we need to build a base model using pre-trained weights. After building a base model, all of the layers of the base model need to be frozen. Freezing layers is an important step as it is a way of controlling the way the weights are updated. By freezing layers, the weights cannot be updated further and thus decrease the computational time for training while minimizing the accuracy loss.



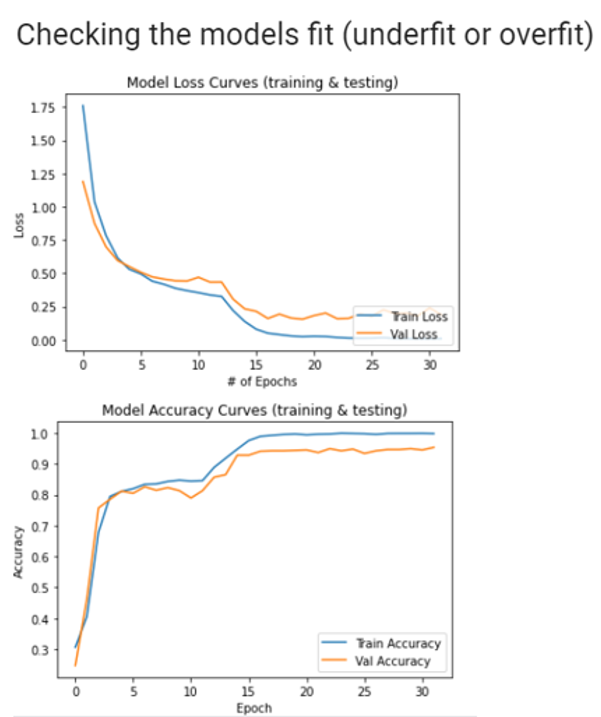
**Figure 5: building base model with VGG-16**

After freezing the base layers, a CNN model can be added and compiled. Since the objective of the project is to classify images, ‘categorical\_crossentropy’ is used for the loss function and ‘adam’ for the optimizer. Hence, the transfer learning model is built now.



**Figure 6:VGG-16 model architecture**

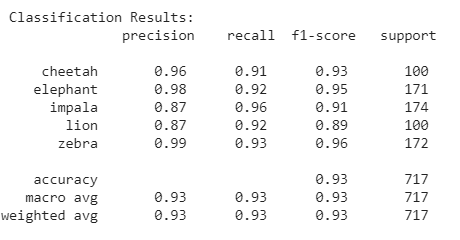
Once the transfer learning model is built, the model then needs to be trained on the training data and the validation data. The validation data is used for training the model in order to give validation accuracy and validation loss, which are useful for comparing against training loss and training accuracy in order to know if the model is underfitting or overfitting.



**Figure 7: VGG-16 model loss & accuracy curves**

The next step is to test the model to make predictions. The testing data is the data that is unseen by the model during training. During model training, the model is only exposed to the training data and validation data. The testing data is hidden until model testing is to see how the model makes predictions on the dataset that the model has never seen before. This helps understand how good the model’s generalization ability is. One of the most important steps in evaluating model performance is to look at additional performance metrics such as F1-score, Precision, Recall, Support, and ROC-AUC graphs due to the imbalanced dataset.

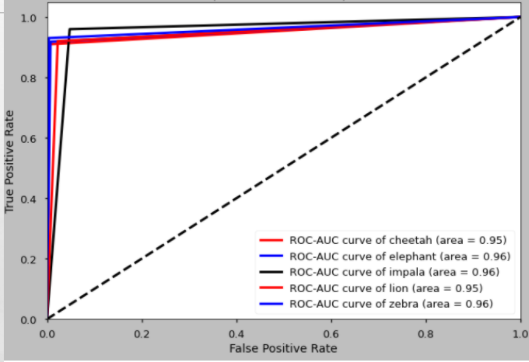
Precision (TP/(TP+FP)) is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. Recall (TP/(TP+FN)), also called sensitivity, is the ratio correctly predicted positive observations to all observations in an actual class. F1-score is the weighted average of Precision and Recall. As F1-score takes both FPs and FNs into account, it can be more difficult to understand, but is usually more useful than mere accuracy. Lastly, Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.



**Figure 8: classification results of VGG-16**

As shown in Figure 8 above, the classification report suggests strong numbers in all categories. First, F1-score ranges from 0.89 to 0.96 and the accuracy is 0.93. Precision also ranges from 0.87 to 0.99, which suggests that in most cases, the model correctly predicts positive observations to the total predicted positive observations.

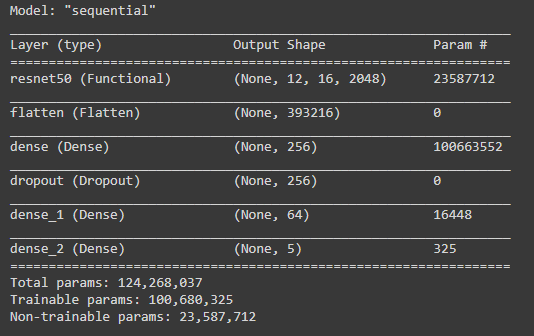
The last step is to look at the ROC graphs for the model. The lowest AUC value is (AUC=0.95) for the cheetah and the lion classes while the highest AUC value is (AUC = 0.96) for the elephant, impala, and zebra classes. The low AUC values make sense because the support for the cheetah and lion classes = 10 indicating that there are 100 images/class in the testing folder. Furthermore, the high AUC values for the elephant, impala and the zebra classes make sense because the support for them is around 170 images indicating that there are 170 images/class in the testing folder.

****

**Figure 9: ROC curves of VGG-16**

**4.2 ResNet50**

ResNet50 is another pre-trained CNN of 50 layers deep used in this project. ResNet is a short name for Residual Network that supports residual learning. ResNet50 has proven its ability by winning the ImageNet challenge in 2015. Figure 10 below shows the model architecture of ResNet 50.



**Figure 10: architecture of ResNet50**

Once the model is built, it is ready to be compiled and trained. ResNet50 also uses ‘categorical\_crossentropy’ for the loss function and ‘nadam’ for the optimizer.

Chart, histogram

Description automatically generated

**Figure 11: ResNet 50 accuracy curve**

**Table

Description automatically generated**

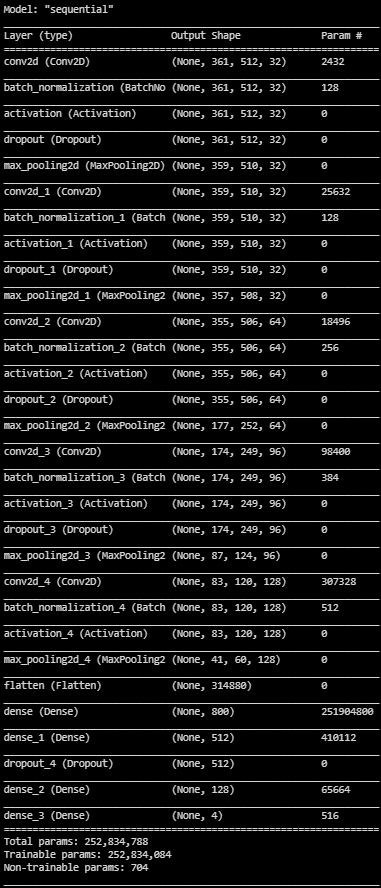
**Figure 12: ResNet 50 classification report**

As shown in Figure 11 and 12 above, the ResNet50 model shows positive classification results. First, F1-score ranges from 0.88 to 0.99 and the accuracy is 0.96. Precision ranges from 0.78 to 1.00, which suggests that in most cases, the model correctly predicts positive observations to the total predicted positive observations.

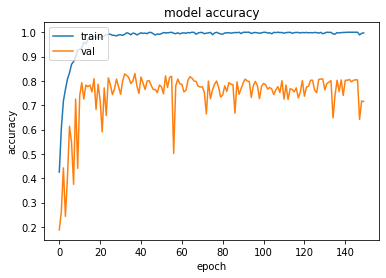
**4.3 CNN from Scratch**

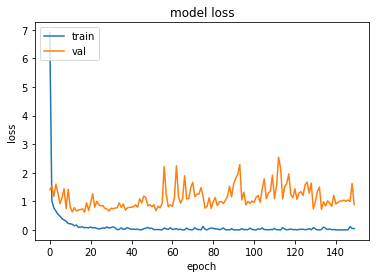
To act as a baseline for our transfer models, we also built a CNN from scratch and attempted to classify the results. After realizing that our CNN built from scratch was unable to classify all 5 images in a suitable amount of time without being shut down by colab, we settled on attempting only 4 images not including the male and female lions. Even with the reduced number of classifications, the scratch CNN was unable to classify on the test data even with very high accuracy and low results on the training data.

Below is the architecture of the scratch CNN model created.



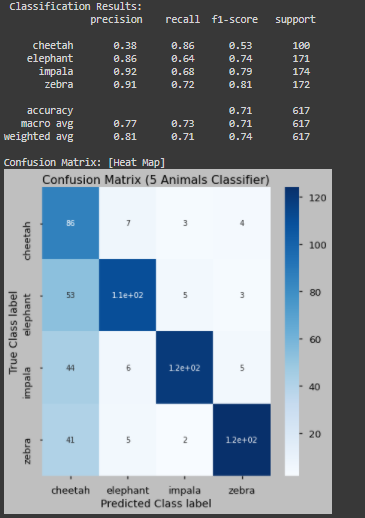
**Figure 12: CNN Architecture**

 **Figure 12: Accuracy of Scratch CNN Over Epochs**

****

**Figure 13: Loss of Scratch CNN Over Epochs**

As illustrated above, it’s clear the scratch CNN reached a peak ability to classify the unseen validation data. No matter how many epochs were run, the model was never able to improve on unseen data, and we saw similar behavior when looking into the unseed test data as well.



**Figure 14: Scratch CNNResults**

As illustrated above, the model struggled at accurately predicting the results when exposed to new data. While having relatively high precision for zebra, elephant, and impala, the model clearly struggled with the cheetah thinking many more pictures were of cheetah than there should have been. This may be a result of the model being unable to identify necessary features as well as the pretrained models.

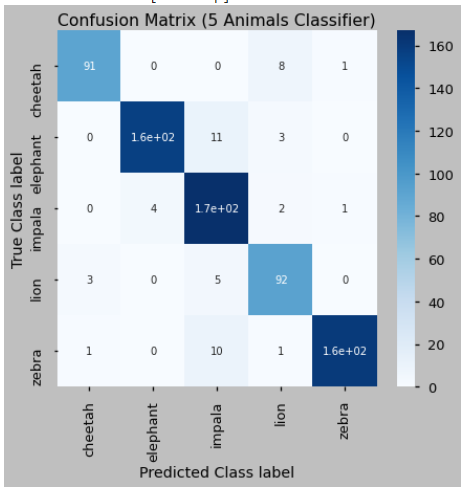
Giving the difficulty the model had with predicting even 4 classes, the decision was made to no longer consider this model in our final results. It was helpful to provide a baseline and it was exciting to achieve even these modest results, however our scratch CNN cannot compete with the resources and total volume of training data available to the pretrained models.

**5 Model Performance**

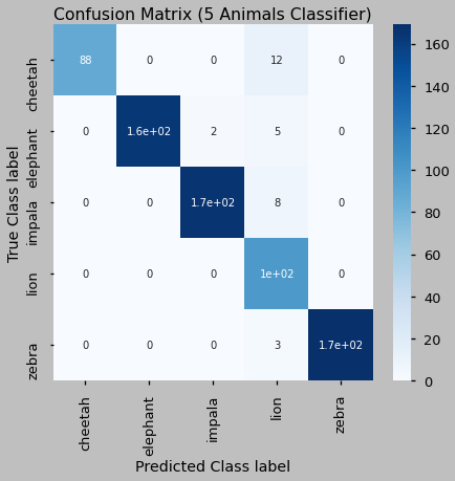
|  |  |  |  |
| --- | --- | --- | --- |
| **Accuracy** | **VGG-16** | **ResNet50** | **Scratch CNN** |
| train | 0.99 | 0.91 | 0.99 |
| test | 0.93 | 0.96 | 0.75 |
| validation | 0.94 | 0.98 | 0.74 |

**Table 1: Accuracy table of 3 models**

The VGG-16 model’s average F1-score is 0.93 while ResNet50 model’s average F1-score is 0.95. The VGG-16 model struggles in predicting impalas while ResNet50 struggles in predicting lions.

****

**Figure 15: VGG-16 confusion matrix**

****

**Figure 16: ResNet 50 confusion matrix**

**7 Conclusion**

Room for improvement includes finding a way to handle blank images, visualizing the neural networks computation using Tensor Flow.

Furthermore, both the transfer learning models based on pre-trained CNN models ResNet50 and VGG-16 had an easy time identifying the zebra, elephant and the cheetah classes while overpredicted the lion and impala classes.

Last, due to the VGG-16 based transfer learning model having a 4% accuracy mismatch between training and validation (indicates an ideally fit model) in comparison to ResNet50 which has a 7% we believe that the VGG-16 is the winning model for scaling up to more larger and unbalanced datasets.

**REFERENCES**

[1]TensorFlow. 2020. *Transfer Learning And Fine-Tuning | Tensorflow Core*. [online]Available at: <https://www.tensorflow.org/tutorials/images/transfer\_learning> [Accessed 20 November 2020].

[2]Ram Sagar. 2020. What Does Freezing A Layer Mean And How Does It Help In Fine Tuning Neural Networks. (October 2020). Retrieved December 5, 2020 from https://analyticsindiamag.com/what-does-freezing-a-layer-mean-and-how-does-it-help-in-fine-tuning-neural-networks/

[3]Renuka Joshi Solutions and \* Name. 2016. Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. (November 2016). Retrieved December5,2020fromhttps://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

[4]The scikit-yb developers. Classification Report¶. Retrieved December 5, 2020 from https://www.scikit-yb.org/en/latest/api/classifier/classification\_report.html