Aerial Scene Classification by Convolutional Neural Networks

Aviral Bhatnagar

I. INTRODUCTION

This project was inspired from Marco et. al [1] that carried out land use classification of remote sensing images by Convolutional Neural Networks. The primary objective of this project is to compare the performance of two CNN architectures, Inception V4 and VGG16 in aerial scene classification, with two different learning modalities. Besides extracting features of a pertained model, I have fine-tuned the top layers of both the architectures using new datasets. I have carried out experiments on three datasets, with markedly different characteristics. Artificial neural networks are inspired from models of the biological brain, and try to reproduce some of its functions by using interconnected processing units. Considering the subtle differences in characteristics among categories in scene classification, deep learning is considered far more superior than pixel-based or object based approaches. For instance, land covers characterizing a given class may present a large intra-class variability coupled with low inter-class distance. Deep learning models are therefore ideal to pick these nuances. I have attempted to demonstrate how different CNN architectures compare with each other in scene classification. For this, I have used UC Merced Land Use dataset, Brazilian Coffee Scenes dataset, and Keras's built-in CIFAR10 dataset. In the subsequent sections, I have explained the architectures and the datasets in more detail. I concluded this project with my results, the challenges I faced and potential scope of improvement in the results in future work. Finally, I added a section explaining what I learned and what personally made a significant impact on my overall experience.

II. RELATED WORK

Marco et. al [1] have used GoogleNet and CaffeNet architectures for land use classification. The paper utilizes three modalities, learning from scratch, feature extraction and fine-tuning. They used two datasets, UC Merced Land Use and Brazilian Coffee Scenes. earlier, I had intended to replicate the paper myself using Inception V1 (GoogleNet) and Inception V4 models. However, I soon realized its narrow scope and decided to make a few architectural modifications. The paper carries out the experiments on a notebook equipped with an NVIDIA GeForce GT 750M 2048 MB GPU. On *UC-Merced LandUse* dataset, the results achieved by the paper have been tabulated below:

CNN	Design	Iterations	Accuracy
CaffeNet	from scratch	100,000	85.71
	fine-tuning	20,000	95.48
	feature vector	5,000	94.28
GoogLeNet	from scratch	100,000	92.86
	fine-tuning	20,000	**97.10**
	feature vector	5,000	94.38

We can notice that the accuracies aren't significantly different each other based on different architectures and modalities. However, my own experiments yielded results that were significantly different from each other. We can notice a similar phenomenon with the *Brazilian Coffee Scenes* dataset:

CNN	Design	Iterations	Accuracy
CaffeNet	from scratch	20,000	90.17
	fine-tuning	10,000	90.94
	feature vector	5,000	85.02
GoogLeNet	from scratch	20,000	**91.83**
	fine-tuning	10,000	90.75
	feature vector	5,000	84.02

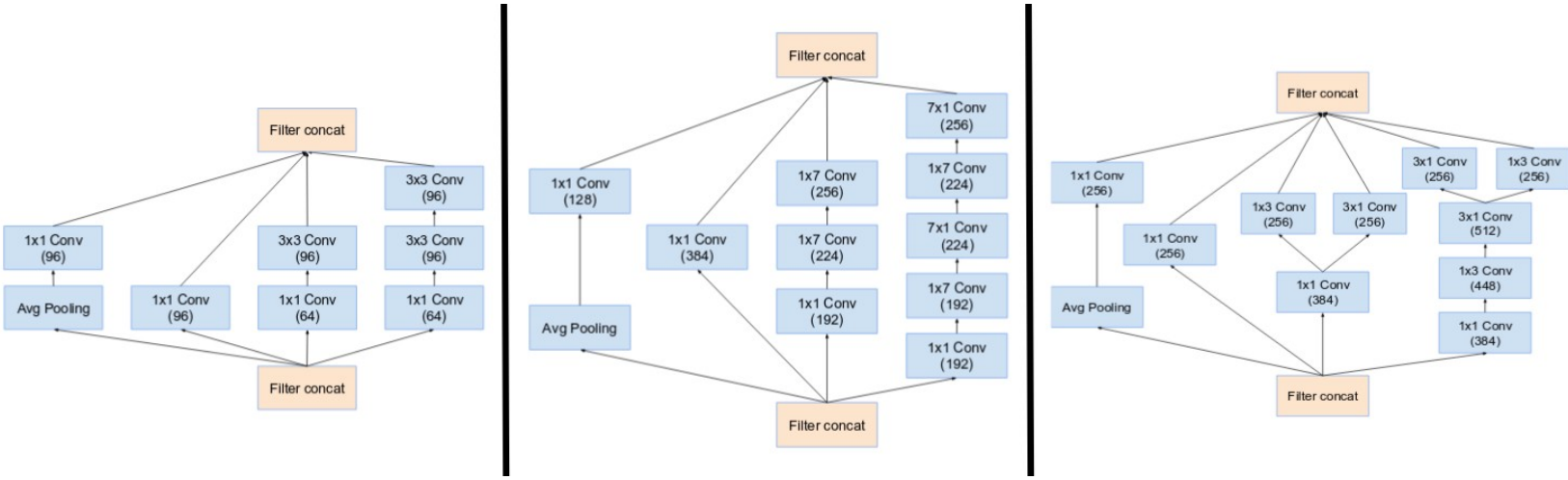
TABLE III

III. ARCHITECTURES USED

1. V*GG-16:* The VGG network architecture was introduced in 2014 by Oxford's renowned Visual Geometry Group (VGG) which achieved laudable performance on the imagine dataset. It is characterized by its simplicity when compared to Inception-v4. It uses on 3x3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handles by max pooling[2]. The max pooling layers are of size 2x2. The 16 stands for the number of weight layers in the network. In order to make the training easier, a few different versions of VGG 16 and VGG 19 were introduced, as shown in the table below:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

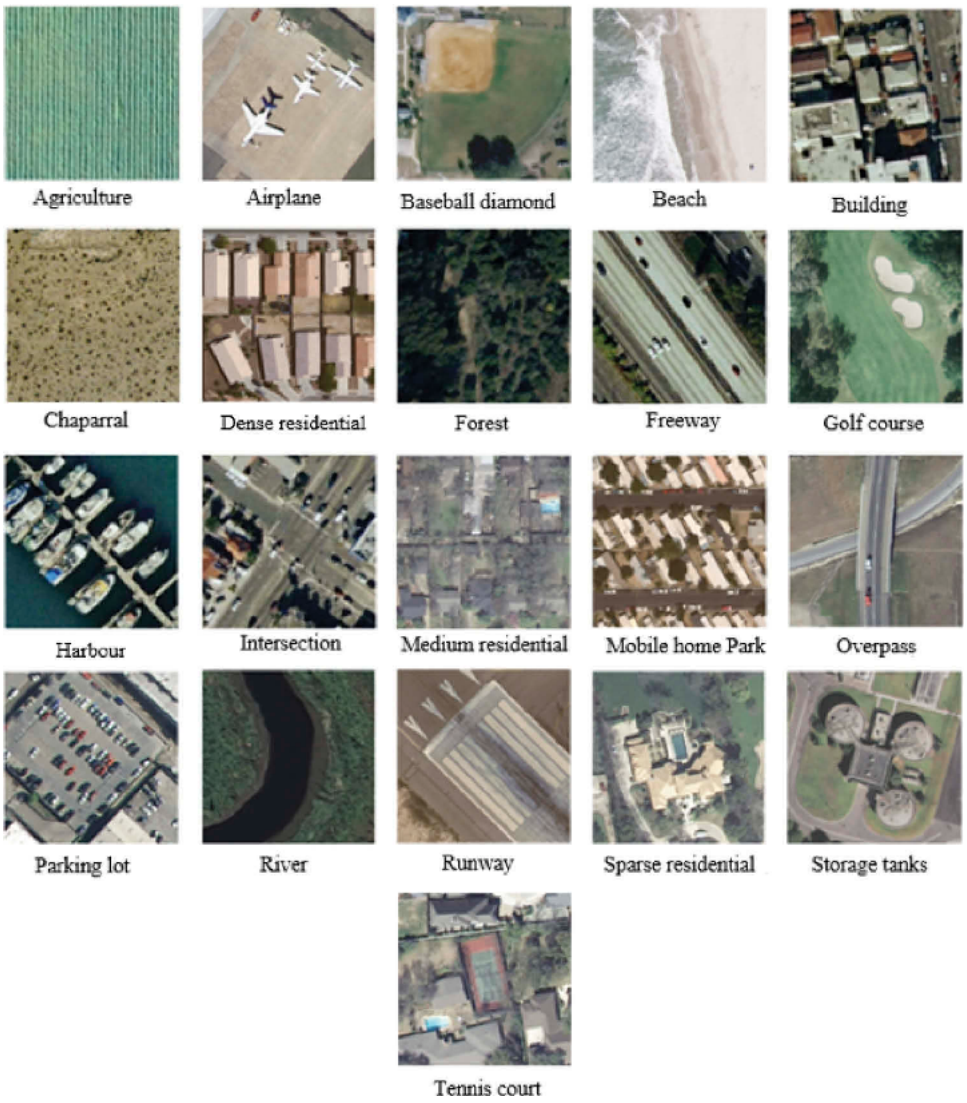
2. *Inception V4*: It is the latest version of the Inception architecture series. It is 22 layers deep (27 including the pooling layers). It has evolved from GoogleNet (Inception-v1) and has a more uniform simplified architecture than the previous versions. Salient parts of an image, the ones responsible for differentiating it from other images, can have varying sizes. Due to this, we would require kernels of varying sizes to capture these features. As a result, it is appropriate to add filters of varying sizes in the model. However, very deep neural networks are prone to overfitting and operations on them are computationally expensive. Google came up with a solution of having filters with multiple sizes on the same level. essentially, this would make the network wider rather than deeper. Inception-v4 combines the upgrades of the earlier versions such as factorizing 5x5 convolution to two 3x3 convolutions for enhanced speed, adding RMSprop Optimizer, factorizing 7x7 convolutions, performing Batch Normalization in the AuxiliaryClassifiers etc. Additionally, it was noted that some of the modules in the architect turn were more complicated than necessary, that resulted in computational bottlenecks [3]. As a solution, the "stem" of the model was modified to introduced "Reduction Blocks" in the stem which change the width and height of the grid. The image shown below depicts the there main inception modules incorporated in the architecture:



IV DATASETS

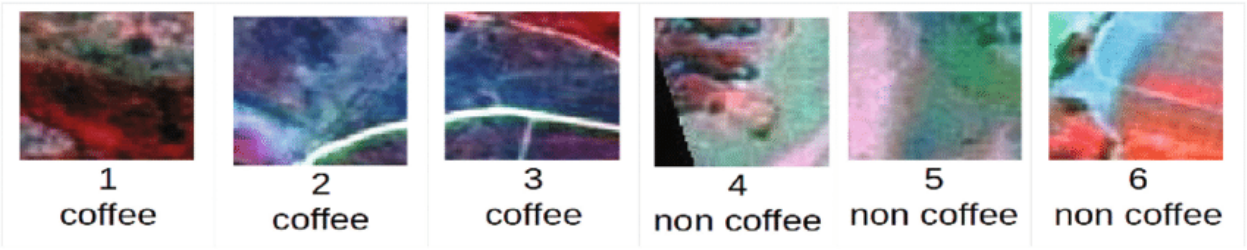
1. *Imagenet:* It is an image dataset organized according to the WordNet hierarchy. It contains more than 20,000 categories with each category consisting of several hundred images. All the contemporary deep learning CNN models like the Inception and VGG series haver been pre-trained on this dataset and readily available for use. This dataset is part of the visual object recognition software research. Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual recognition challenge (ILSRVC), where software programs compete to correctly classify and detect object and scenes [4].

2. *UC Merced Land Use dataset: T*his dataset was released in 2010 and is extracted from large optical images of the US Geological Survey, taken over various regions of the US. **2100 256x256-pixel** images are selected and labelled as belonging to one of **21 land use classes, 100** for each class. They have a high resolution of 30cm and these images share many low level features with general-purpose optical images making them good candidates for fine-tuning pertained CNNs [5]. The images below depict a few examples from the dataset.

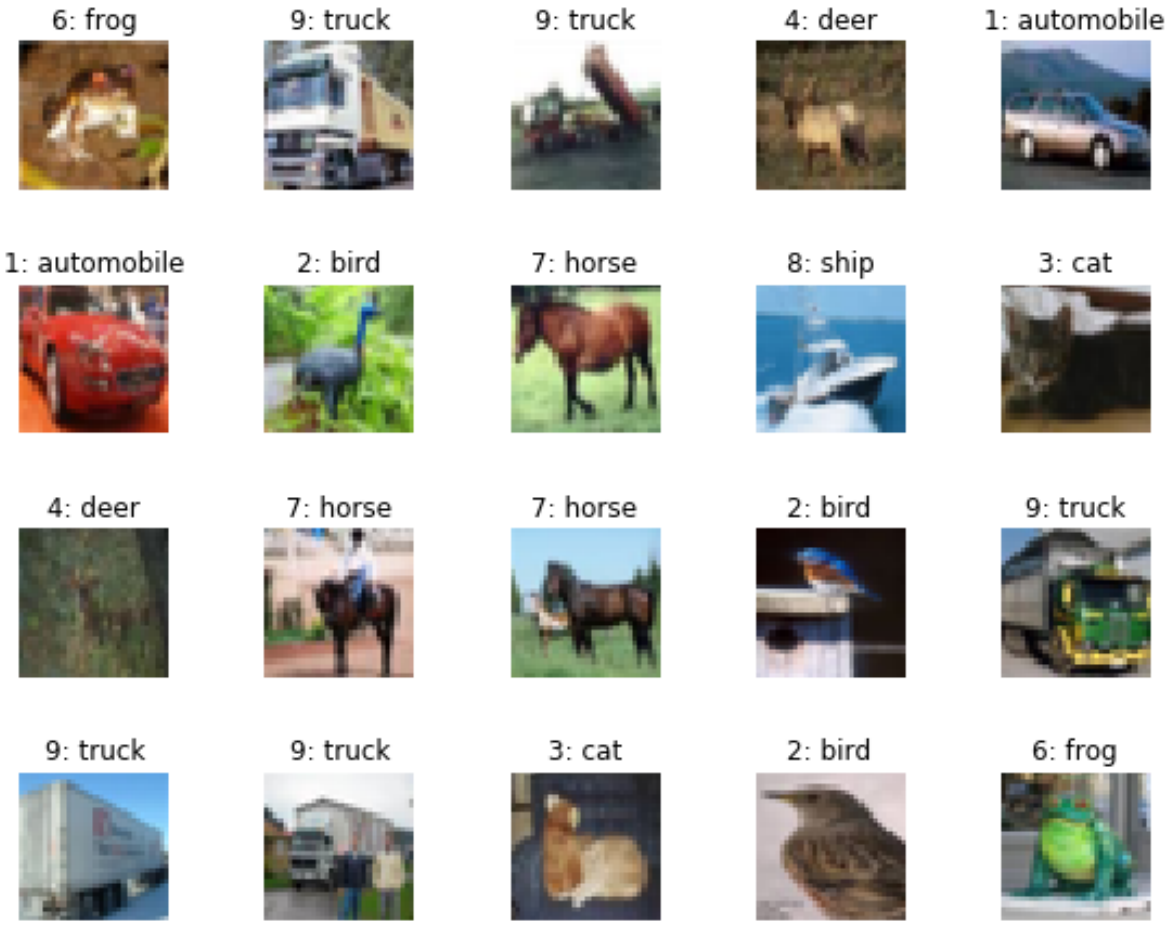


3. *Brazilian Coffee Scenes dataset:* This dataset is a composition of scenes taken by SPOT sensor in 2005 over four counties in the State of Mines Gerais, Brazil: Arceburgo, Guaranesia, Guaxupé and Monte Santo. It has many intraclass variance caused by different crop management techniques. The whole image set of each country was partitioned into multiple tiles of **64 x 64** pixels. The identification of coffee crops (i.e. ground-truth annotation) was performed manually by agricultural researches. The dataset has non-coffee (less than 10% coffee pixels) and coffee

(at least 85% coffee pixels) classes. It is divided into 4 folds of 600 images each and the 5th fold has 476. Before running the model on this dataset, I combined all the images along with their labels in one folder [6]. the image below shows a few examples of the images from the dataset.



4. *CIFAR10 dataset:* This dataset consists of **60000 32x32 color images in 10 classes**, with **6000 images per class**. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class [7]. The images below are a few example from the dataset:



V. EXPERIMENTAL RESULTS

The experiments below could not have been possible without this GitHub repository [7].I carried out a number of experiments in an attempt to conclusively prove the superiority of one model against the other. My assumptions before the project were completely in contrast to the results I got. I inspected two different learning modalities for each architecture across 3 datasets. I modeled 12 different combinations of architectures, not counting the hyperparameter modifications for each model. The modalities I used were:

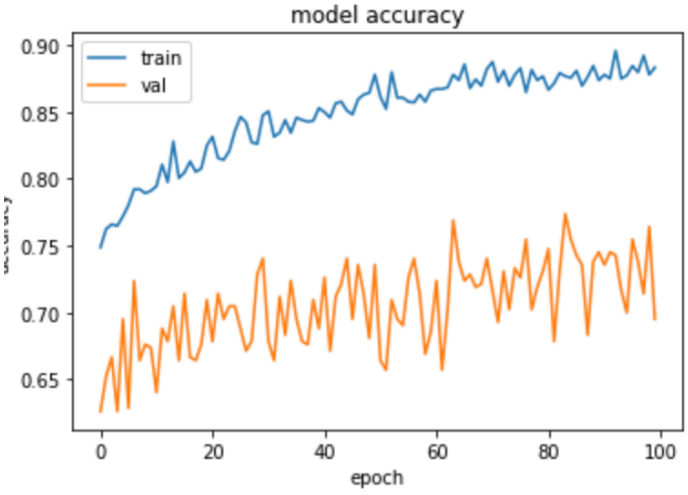
- *Feature Vector*: The output of the penultimate layer of a pre-trained net is used as a feature vector for classification.
- *Fine Tuning:* A pertained net is used, and I trained the last layer with the dataset at hand, along with adding Zero Padding layers.

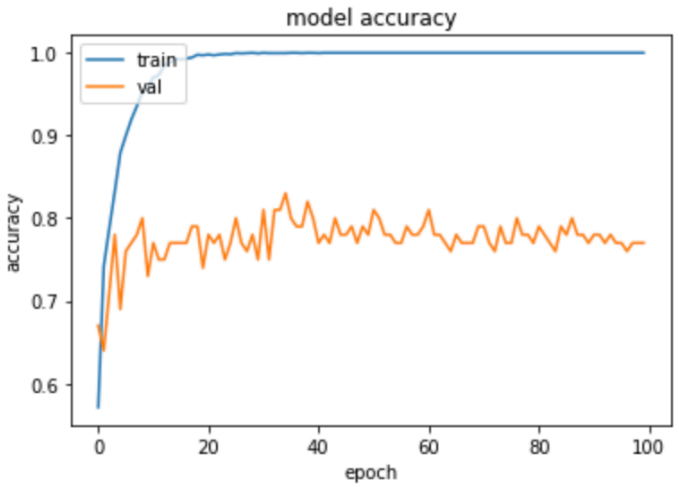
In an attempt to keep the comparisons fair, I kept the specifications of my hyperparameters as consistent as possible across all the models. I had to tweak them every now and then to improve the classification performance. Following are the specifications:

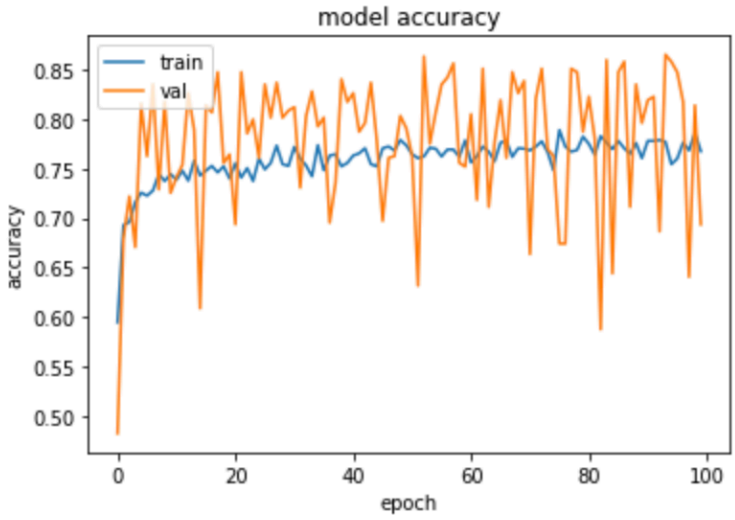
1. *Optimizer*: Rmsprop; SGD in some cases with *lr = 1e-2* and *momentum =0.9*
2. *Batch Size:* 10
3. *Epochs:* 100 for VGG16 and 50 for some cases with Inception-v4.
4. *Dropout Values :* 0.5.
5. *Number of layers trained in Fine-tuning*: 1

VGG16:

Feature Vector:

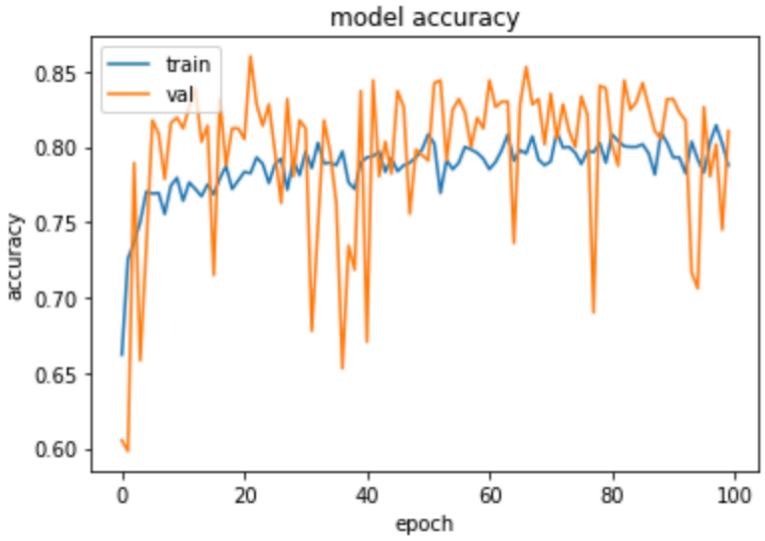






UC Merced Land Use

Brazilian Coffee Scenes (RMS prop).



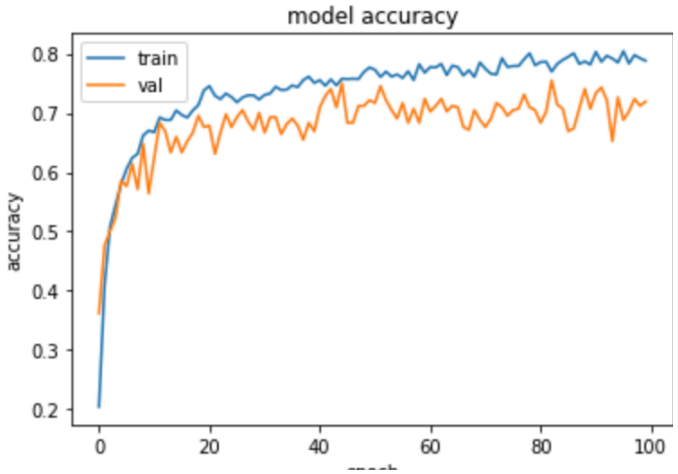
CIFAR-10

Brazilian Coffee Scenes(SGD)

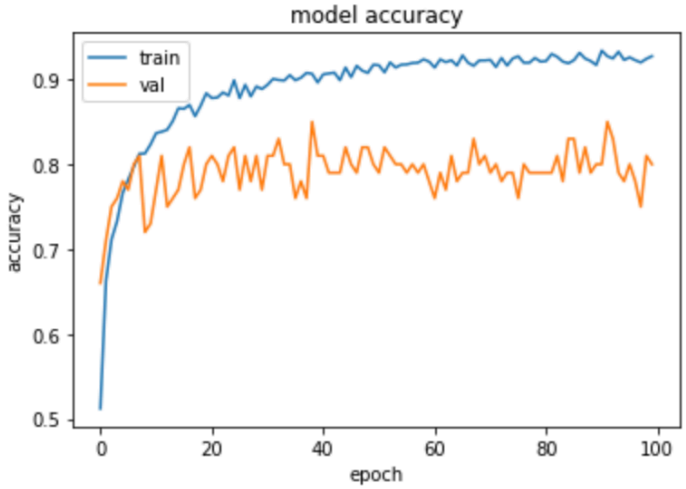
Observations::

1. My model performs considerably well on UC Merced dataset, given the fact that it contains 21 classes. It achieves an accuracy of 69.6%.
2. The model on CIFAR-10 overfits the data, giving an accuracy of 77%. It contains 10 classes, so an impressive accuracy. I attempted to tackle overfitting in the fine tuning section.
3. My accuracy on Brazilian coffee scenes using both the optimizers is not very impressive, considering the fact that the dataset contained only 2 classes. Before starting the project, I had expected this dataset to perform poorly given its mediocre resolution. Moreover, the validation graph is extremely erratic. I tried to solve this problem by adding momentum to the gradient descent, but to know avail.

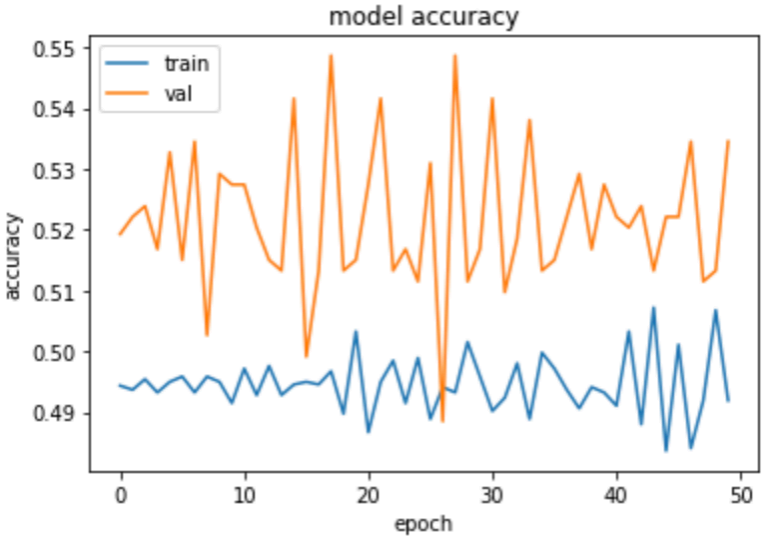
Fine-Tuning:



UC Merced Land Use



CIFAR-10



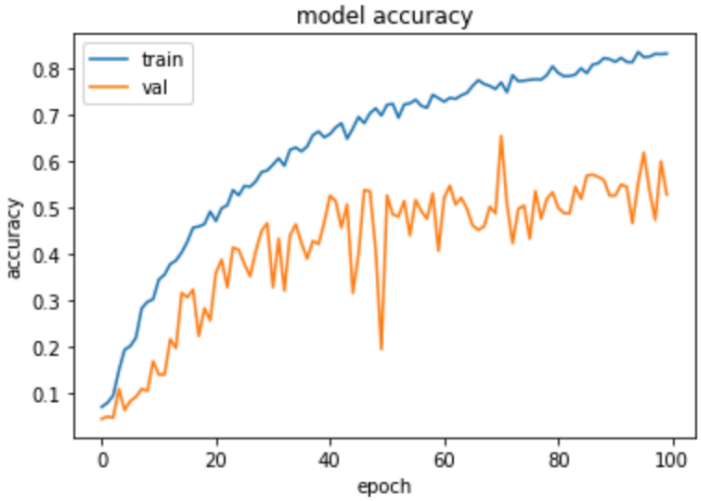
Brazilian Coffee Scenes(RMSprop)

Observations:

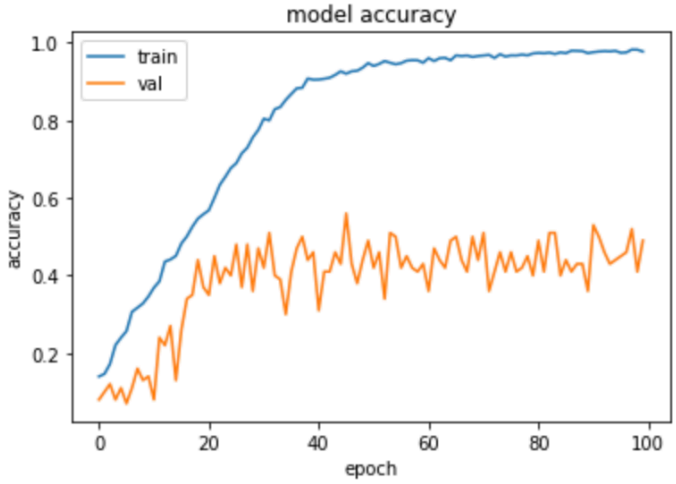
1. We can clearly notice that fine-tuning the last layer and adding a dropout layer of value 0.5 enhanced the performance on the UC Merced LandUse dataset. The accuracy is now 73.1%.
2. Fine-tuning with the same hyper parameters also reduced overfitting in CIFAR-10, thereby enhancing the accuracy to 80%.
3. Shockingly, fine-tuning the model on Brazilian Coffee Scenes deteriorated the performance significantly (accuracy not being better than a random classification). My understanding is that the validation dataset I chose is under representative of the dataset for such a complex model.

Inception-V4:

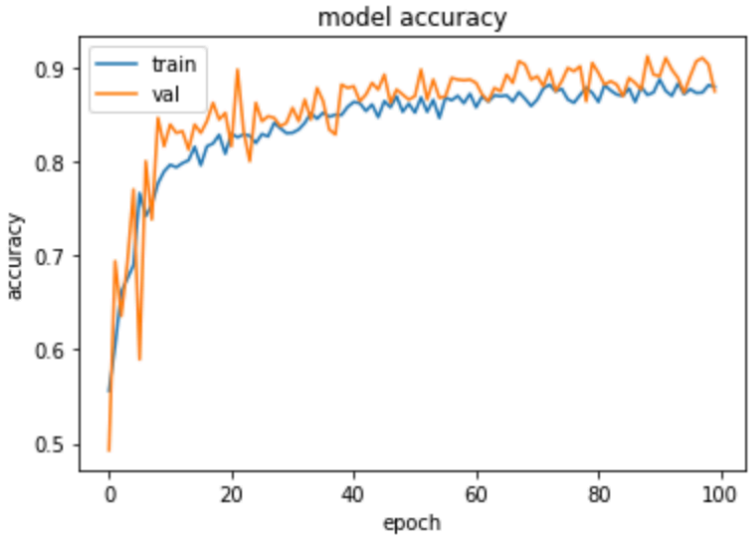
Feature-Vector:



UC Merced LandUse



CIFAR-10

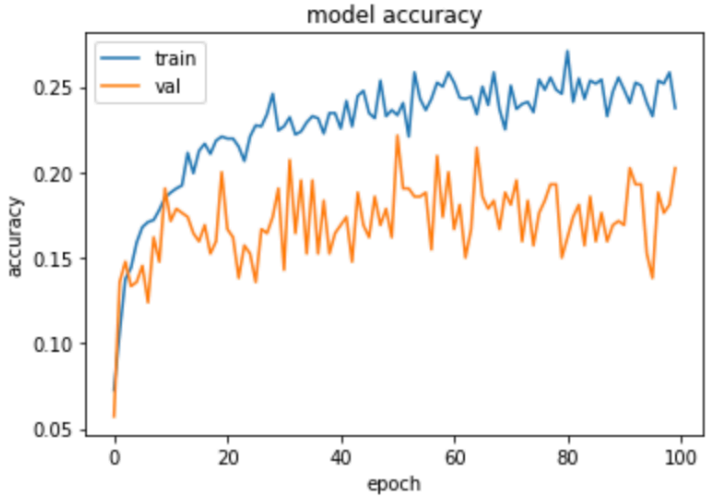


Brazilian Coffee Scenes(RMSProp)

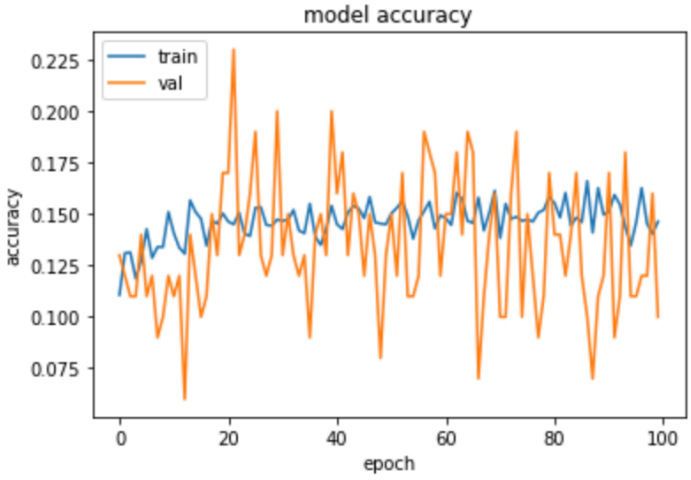
Observations:

1. In spite of Inception-V4 being considered a more advanced model than VGG16, its performance pales in comparison to the latter across most datasets.
2. Surprisingly though, the performance on Brazilian Coffee Scenes significantly improved.

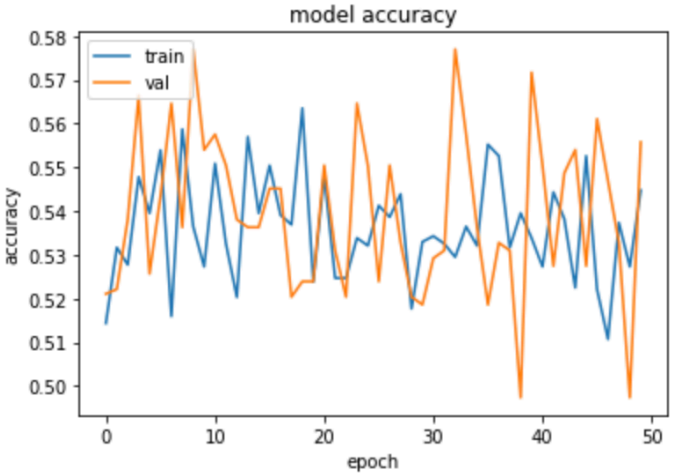
Fine-Tuning:



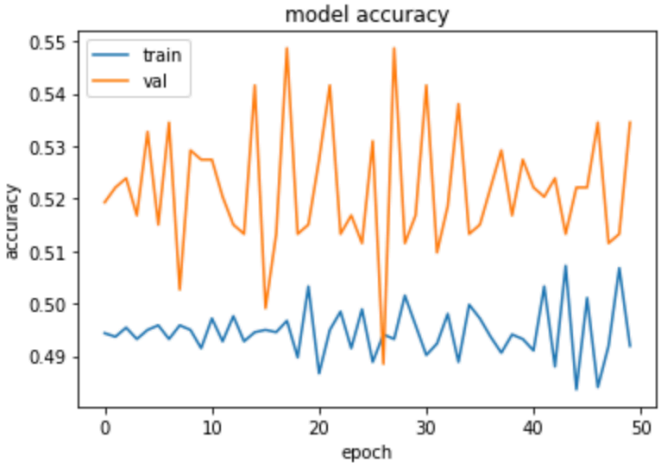
UC Merced Land Use



CIFAR-10



Brazilian Coffee Scenes(SGD) (RMSProp)



Brazilian Coffee Scenes

Obserbvations:

1. This results after fine-tuning Inception-V4 have been abysmal overall. Training the last few layers seems to have worsened the learning ability of the model.
2. My understanding is that in spite of data augmentation, the amount of data at hand was not sufficient to train the model, thereby making the results extremely erratic.

RESULT SUMMARY:

	Feature Vector			Fine-Tuning		
	UC Merced Land Use	CIFAR-10	Brazilian Coffee Scenes	UC Merced Land Use	CIFAR-10	Brazilian Coffee Scenes
VGG16	69.6&	77%	81%(SGD) 69%(RMS)	73.1%	80%	53%
Inception-V4	52%	50%	84%(RMS)	20%	12%	53%(RMS) 55%(SGD)

VI. MY INTERPRETATION OF THE RESULTS

- My assumption that Inception V4 would perform significantly better than VGG16 was flawed. The results were quite the opposite, with VGG 16 performing better in many cases.
- Even though Inception-V4 does not perform as good, I feel it has a better scope of improvement due to its slanting graph.
- More research needs to be done into why my fine-tuning on Inception model performed so poorly. My understanding is that the validation dataset is unrepresentative for such a complex model.
- My assumption initially was that Brazilian Coffee Scenes would perform poorly, given its low resolution. That's exactly what happened. however, Inception-V4 significantly improved the performance, perhaps due to its better ability to handle overfitting.
- Not only does the complexity of the model matter, but many other factors play a role in the performance of a model, hence fine-tuning is very important.
- Fine-tuning a complex model may not necessarily enhance the performance than just a mere feature extraction.

VII. CHALLENGES

Easily the most significant challenge I faced while engineering this project was my remarkable computational limitations. I used my personal MacBook Pro and ran my models on Google Colab. Even though Colab provides a **GPU of 1xTesla K80 , having 2496 CUDA cores, compute 3.7 and 12GBb (11.439GB Usable) GDDR5 VRAM**, the complex deep learning models I attempted to implement proved to be too overwheMY lming for the system. It would take more than half a day to fine tune both VGG 16 and Inception models with 50 epochs. Moreover, google Colab has an idle timeout of 90 minutes and absolute timeout of 12 hours. As a result, I could not run my models overnight and was severely restricted to daytime execution. Running multiple models parallelly further made the computation even slower. I would have to wait for 6 hours to get tangible results and make the amends accordingly.

The second most significant challenge I faced was the lack of explicit specifications of the hyperparameters involved in implementating the deep learning architectures. The most intricate nuance like the learning rate, decay rate, decay algorithm used, steps per epochs, train/validation split, data augmentation specifications etc all can significantly change the performance of the model. And adding to that the slow processing time of each model made tweaking them even more difficult.

Lastly, the price that all of us need to pay for being at the forefront of cutting edge research like Computer Vision is the lack of proper documentation of the subject. Rectifying dimensional and logical errors in my code was no easy feat. Excessive dependence on resources like ***stackoverflow.com*** made the project more convoluting. I spent more than a month to get my first model to work.

VII. WHAT I LEARNED

The idea of giving a computer the ability to comprehend what it sees through a camera has always fascinated me. During the beginning of the course, my knowledge of deep learning, let alone convolutional neural networks was very shallow and vague. This project not only made me understand the concepts in theory, but also helped me learn how to apply them in solving real-world problems. I soon realized that what appears straightforward in theory can prove to be a lot more convoluting while implementing it.

An architecture is never universally better across all datasets and circumstance, no matter how cutting edge the authors claim them to be. A lot of factors play a role, and we should fine-tune them accordingly. I understood how critical it is to tweak the model with utmost precision to achieve optimal results. I learned about one significant problem with deep neural networks. It is that they have a lot of hyperparameters to tune and very small solution space. Thus, finding good ones is more like an art rather than an engineering task. I understood the meaning behind several hyper parameters like learning rate, steps per epoch, learning algorithms like Stochastic Gradient Descent, Rmsprop. I learned how I can further prevent a model from overfitting on the training set by fine-tuning it by adding Dropout and Batch Normalization layers, and by modifying the data through Data Augmentation. I understood the intricate details behind the Inception v4 and the VGG16 models.

Most importantly, I learned how substantially important access to cutting edge computational resources are. My project would not have taken the kind of time it took for completion and I might also have tried a few different algorithms to further improve my results. But my scope was heavily limited by the resources at hand. But on the brighter side, these are the very reasons that spiked my curiosity in the field and made me contemplate doing a PhD. For that, I am forever grateful to this experience.

IX. REFERENCES

[1] Land Use Classification in Remote Sensing Images by Convolutional Neural Networks- Marco et al.

[2] https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

[3] https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202
[4] https://en.wikipedia.org/wiki/ImageNet

[5] Yi Yang and Shawn Newsam, "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010.

[6] O. A. B. Penatti, K. Nogueira, J. A. dos Santos. Do Deep Features Generalize from Everyday Objects to Remote Sensing and Aerial Scenes Domains? In: EarthVision 2015, Boston. IEEE Computer Vision and Pattern Recognition Workshops, 2015.

[7] https://github.com/flyyufelix/cnn_finetune

Aviral Bhatnagar