Weather Prediction Project Report

Introduction:
The problem statement is to recognize the weather condition in the given picture by using extracted useful features from the image.

For some image recognition tasks or computer vision studies, intuition and illumination kinds of difficulties may cause unexpected results. It can be tricky to define weather conditions from a given scene. Even human eyes and brain can be insufficient to recognize weather patterns. That's where an image recognition model can be helpful. Moreover, people may want to organize pictures in their mobile phone based on weather or season. It can also help autonomous cars, alert systems etc.

Even though this problem serves a very wide area to be researched, there are not sufficient publications or projects done in this research area. This makes a good case to pursue research work in this area. I did a comprehensive study of related research work and accomplish weather condition recognition task in a large class range.

Dataset description:
The dataset used in this project has images divided into 5 classes:

1. Cloudy
2. Sunny
3. Rainy
4. Snowy
5. Foggy

The dataset contains a lot more images for cloudy and sunny weather than for the other 3 classes.

Steps followed for weather prediction:

1. Feature Extraction
2. Fitting pre-trained models
3. Training on LeNet5

4. Training on AlexNet

Feature Extraction:

During feature extraction, the following features of the images were extracted:

1. Contrast
2. Brightness
3. Haze
4. Colour histogram
5. Intensity
6. Sharpness

Model Fitting:

Before fitting any model on the dataset, I used stratified sampling to handle the imbalance between the number of samples of different classes. Basically, stratified sampling ensures that the ratio of number of samples of each class remains the same in the training and test data.

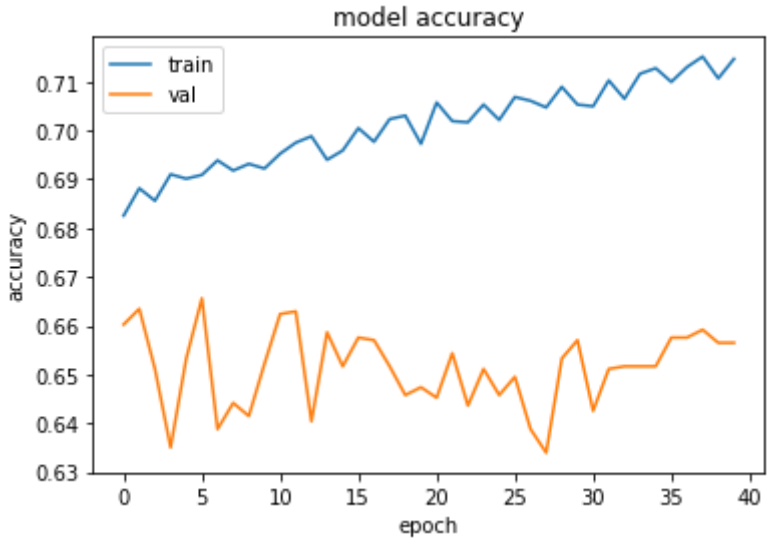
Fitting pre-trained models:

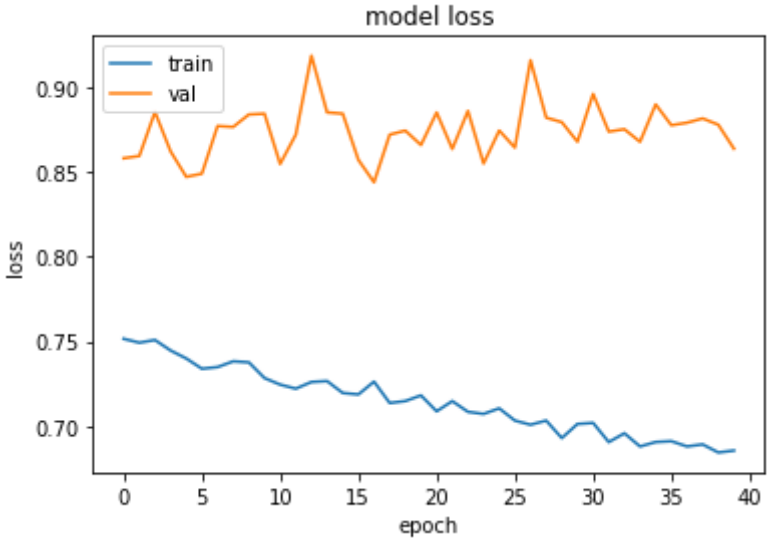
Here, I used the transfer learning approach i.e reusing a model developed for a task on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

I used the following pre-trained models in this project:

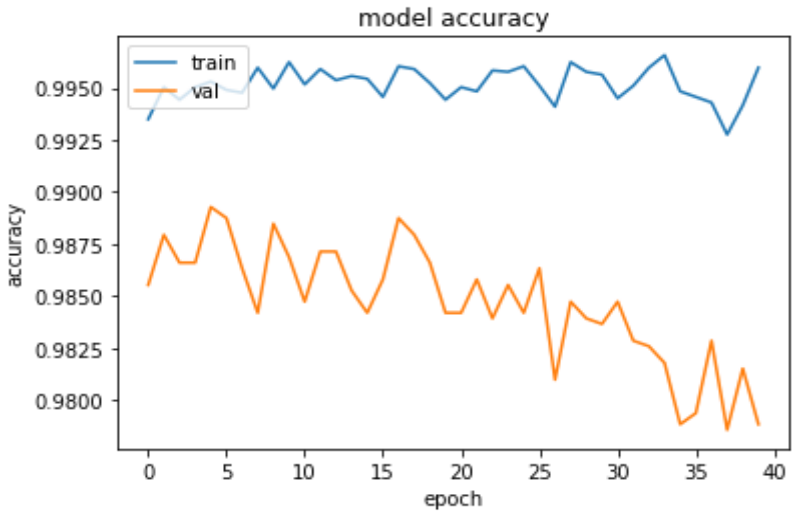
1. **VGG16:** It is the CNN architecture that won the ILSVRC (Imagenet) competition in 2014. Instead of having a large number of hyper-parameters, it has convolution layers of 3x3 filter with a

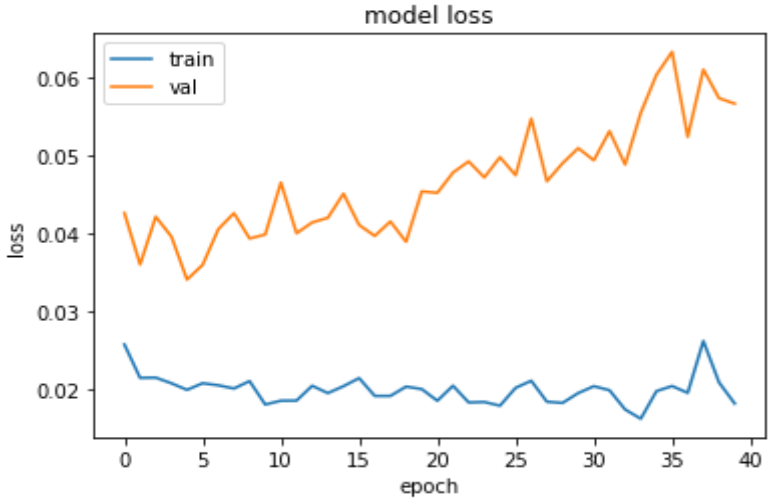
stride 1 and always uses the same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. It has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network has about 138 million (approx) parameters.



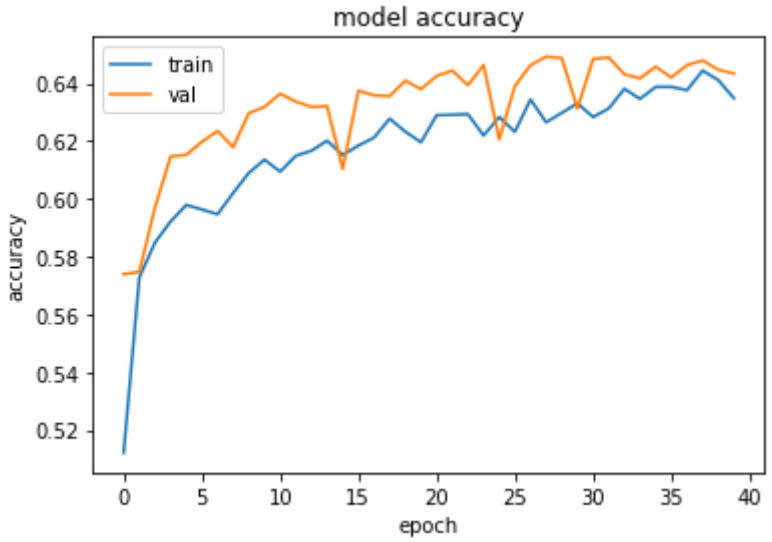


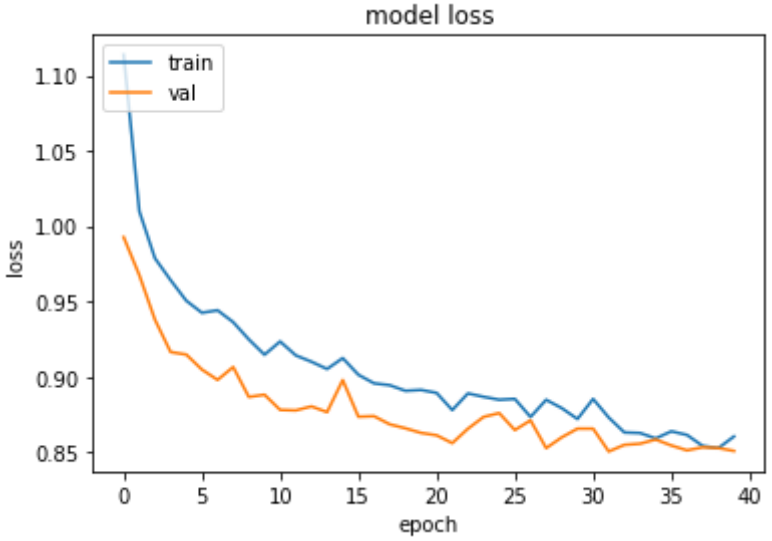
2. **ResNet-50:** ResNet50 is a variant of the ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 floating points operations.





3. **EfficientNet:** EfficientNet-B0 is the baseline network developed by AutoML MNAS, while Efficient-B1 to B7 are obtained by scaling up the baseline network. In particular, EfficientNet-B7 achieves new state-of-the-art 84.4% top-1 / 97.1% top-5 accuracy.





Fitting LeNet5:

LeNet-5:

LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

The first layer is the input layer — this is generally not considered a layer of the network as nothing is learnt in this layer. The input layer is built to take in 32x32, and these are the dimensions of images that are passed into the next layer.

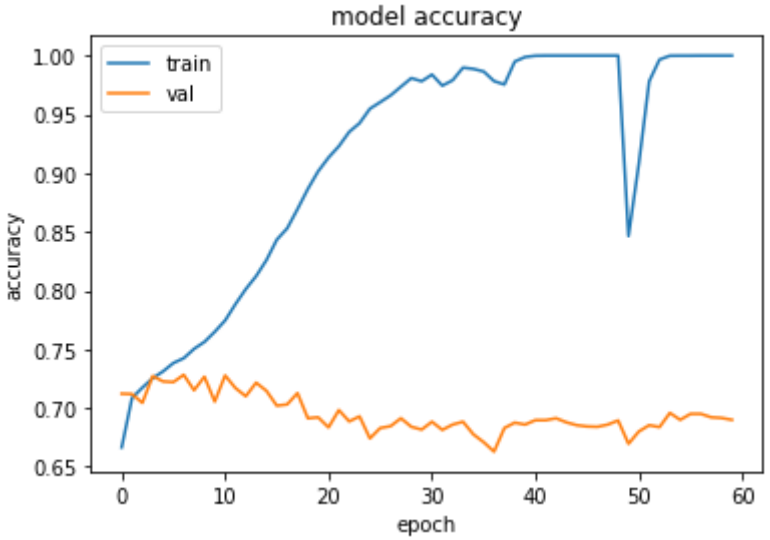
The grayscale images used in the research paper had their pixel values normalized from 0 to 255, to values between -0.1 and 1.175. The reason for normalization is to ensure that the batch of images have a mean of 0 and a standard deviation of 1, the benefits of this is seen in the reduction in the amount of training time. The LeNet-5 architecture utilizes two significant types of layer construct: convolutional layers and subsampling layers.

1. Convolutional layers
2. Sub-sampling layers

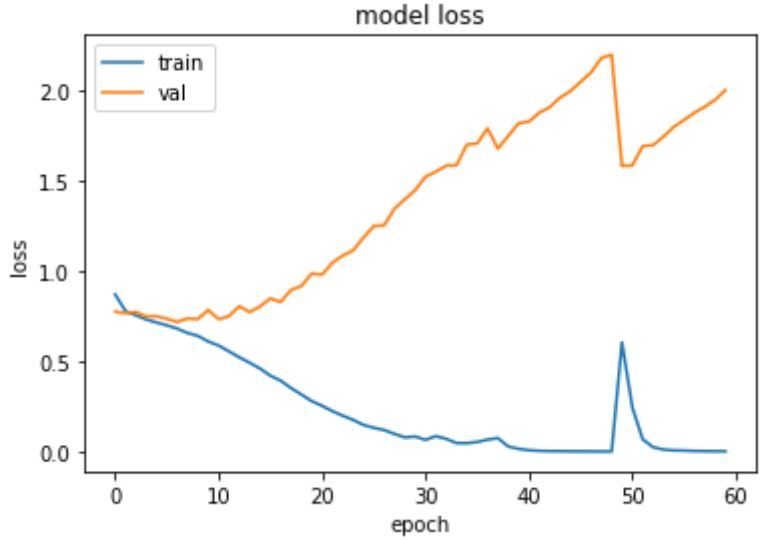
The official first layer convolutional layer C1 produces as output 6 feature maps, and has a kernel size of 5x5. The kernel/filter is the name given to the window that contains the weight values that are utilized during the convolution of the weight values with the input values. 5x5 is also indicative of the local receptive field size of each unit or neuron within a convolutional layer. The dimensions of the six feature maps the first convolution layer produces are 28x28.

A subsampling layer ‘S2’ follows the ‘C1’ layer’. The ‘S2’ layer halves the dimension of the feature maps it receives from the previous layer; this is known commonly as downsampling.

The ‘S2’ layer also produces 6 feature maps, each one corresponding to the feature maps passed as input from the previous layer.



Train-validation accuracy on LeNet-5



Train-validation loss on LeNet-5

Fitting AlexNet:

AlexNet:

The architecture consists of eight layers: five convolutional layers and three fully-connected layers. These are some of the features used that are new approaches to convolutional neural networks:

1. **ReLU Nonlinearity:** AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh.

2. **Multiple GPUs:** AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.

3. **Overlapping Pooling:** CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, models with overlapping pooling generally find it harder to overfit.

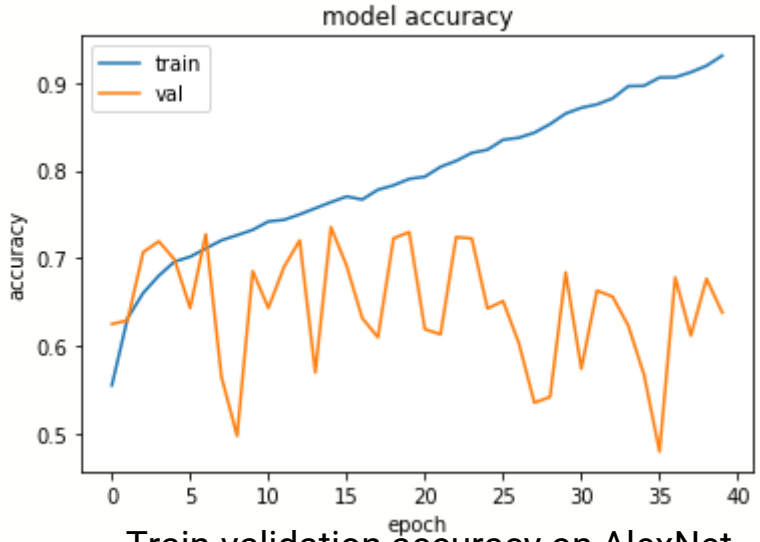
The Overfitting Problem:

AlexNet had 60 million parameters, a major issue in terms of overfitting. Two methods were employed to reduce overfitting:

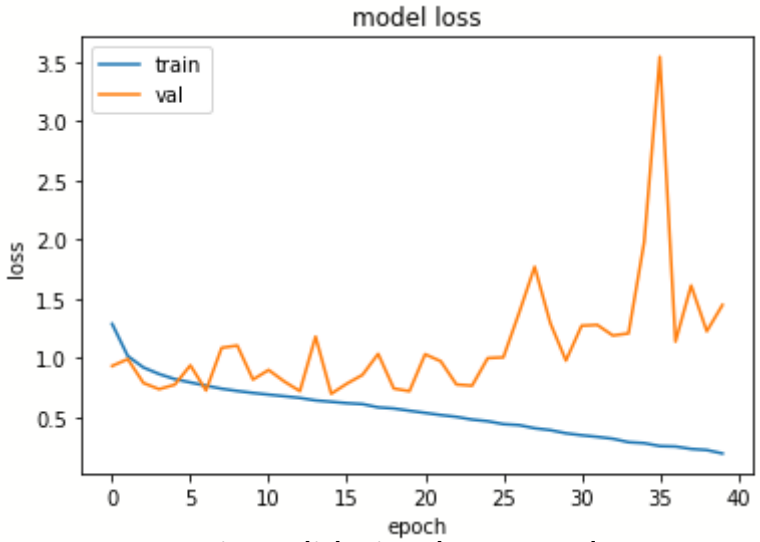
1. **Data Augmentation:** Label-preserving transformation to make data more varied. Specifically, image translations and horizontal reflections are generated, which increase the size of the training set.

2. **Dropout**: This technique consists of "turning off" neurons with a predetermined probability (e.g. 50%). This means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used

with other random neurons. However, dropout also increases the training time needed for the model's convergence.



Train-validation accuracy on AlexNet



Train-Validation loss on AlexNet

Evaluation Metrics:

For evaluating the performance of a machine learning model, the following metrics are generally used:

1. **Accuracy:** It is the ratio of total number of correct predictions made to the total number of predictions made.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2. **Precision** : The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

3. **Recall** : The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

4. **F1 Score**: The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Comparison between different models:

Model Name	Accuracy	F1 Score	Precision	Recall
Pretrained Model: VGG16	0.721	0.590	0.652	0.560
Pretrained Model: Inception	0.709	0.601	0.631	0.579
Pretrained Model: Efficient Net	0.709	0.596	0.634	0.572
LeNet5 Architecture	0.696	0.564	0.586	0.547
Alexnet Architecture	0.721	0.591	0.653	0.560
SVC	0.53	0.53	0.53	0.53
Random Forest	0.54	0.54	0.54	0.55

Conclusion:

After comparing various pre-trained models, standard architecture and other models like SVC and random forests, I found that CNNs performed much better than other models. AlexNet, in particular, gave the best performance among all the CNNs tried.

Reason behind the success of CNN and AlexNet:

Both Support Vector Machines (SVMs) and Convolutional Neural Network (CNN) are supervised machine learning classifiers. Each of them have strengths and weaknesses. A CNN is a parametric classifier that uses hyper-parameters tuning during the training phase. An SVM is a non-parametric classifier that finds a linear vector (if a linear kernel is used) to separate classes. Actually, in terms of the model performance, SVMs are sometimes equivalent to a shallow neural network architecture.

CNN is primarily a good candidate for Image recognition. It is designed to recognize visual patterns directly from pixel images with minimal preprocessing. We should definitely use CNN for sequence data, but they shine in going through huge amounts of image and finding non-linear correlations.Generally, an CNN will outperform an SVM when there is a large number of training instances, however, neither outperforms the other over the full range of problems.

Similarly here also, for the below reasons CNN out perform SVM:

1. Image Dataset - CNN performs well on image predictions.
2. Non-Linearity - SVM is like a shallow CNN, for multi dimensional space and finding non linearity CNN predicts well.
3. Pretrained Model: here we use some pretrained models for CNN which are trained on huge datasets and hence have a better model to segregate classes.

AlexNet: The network had a very similar architecture as LeNet by Yann LeCun but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted of 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and

fully-connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines. And it has 60 million parameters hence its perform very similar to the human brain and able to distinguish different types of weather by processing images.