Intel Image Classification

Project Final Report

ITCS 6156 - Machine Learning

A Project Report
Submitted by :-

Group 16 - MLCode

Name	Niner ID
Apnav Poptani	801137923
Ashesh Shahi	801155354
Pooja Elkal	801137492
Samarth Patel	801132304
Sapna Pareek	801165364

At

University of North Carolina at Charlotte

Introduction

Image classification refers to the labelling of an image into one of a number of predefined classes. It is a widely used technology around the world in every field nowadays. Image classification is one of the major tasks in computer vision investigated for many years [1]. Classifying images are of great interest in many application areas such as image captioning [2], object tracking [3], scene understanding [4], and event detection [5] and for a multitude of other purposes.Many approaches have been taken to solve it and machine learning has become the most promising method to classify images in human accuracy [6].

Now, how do we, humans, recognize a forest as a forest or a mountain as a mountain? We are very good at categorizing scenes based on the semantic representation and object affinity, but we know very little about the processing and encoding of natural scene categories in the human brain. Classification between different objects is a fairly easy task for us as humans, but it has proved to be a complex one for machines and therefore Image Classification has been an important task within the field of machine learning and computer vision.[10]. Well, a similar process we are going to apply in our project "Intel Image Classification".

Problem Statement:

The main aim of this project is to classify the images of natural scenes around the world. We will build a classification model with multiclass output using "Convolutional Neural Network" (CNN) algorithm. We'll use libraries like Scikit Learn, Keras, TensorFlow, etc, for implementation of our classification model.

Motivation:

Why is this important?

This model can be used for applications that use landscape images as their own features, such as cluster recommendation locations that are comparable to user feedback. It can also be used by applications(like Google Photos,apple photos etc) that cluster images from galleries according to scenes like mountains, glaciers etc.

What makes you investigate this problem

Image Classification is a vast growing field.It is being used in every field these days, whether it is Medical Science, Auto-Pilot Car , Electronic devices like phones and laptops. Also, this lets us work on unstructured data, eventually as Machine Learning Engineer or Data Scientist we have to deal with unstructured data in the real world.

Challenges

Our main challenge would be to identify all the images of different categories correctly and build a model that predicts any new image with the correct label from the given categories. Also,if the images are taken in different lighting conditions, at different angles or varying

colors in color images, Image classifiers might get confused with other similar pictures of different categories.

We are using the concept of Data Augmentation which takes the approach of generating additional images from our existing dataset by augmenting them using random transformations that yield similar looking images. It will help models to generalize better and recognize objects in different angles.

Concise summary of solution (approach)

Approach

Classification of Intel Image data, data consist of images of different nature scenes taken around the world. We are building a model that will classify the images on the basis of the scene where the picture is taken. We are using different techniques to correctly classify the images under different categories : (buildings - 0), (forest - 1), (glacier - 2), (mountain - 3), (sea - 4) and (street - 5).

Steps

1. Installation and Configuration
2. Load and Explore the dataset
3. Image Augmentation
4. Build a Model
5. Convolutional Layer
6. Fully Connected Layer
7. CNN Training
8. Validating Accuracy

Dataset

We will be working on the “Intel Image Classification” dataset. The dataset (“Intel Image Classification”) contains images of size 150X150 pixels, which are distributed under six categories. The six categories with their labels are listed as follows: - (buildings - 0), (forest - 1), (glacier - 2), (mountain - 3), (sea - 4) and (street - 5).

We are taking this dataset from Kaggle. But initially this dataset was published on Analytics Vidhya (“https://datahack.analyticsvidhya.com”) by Intel to host a Image Classification Challenge.

Background

Deep Survey of Literature

ImageNet Classification with Deep Convolutional Neural Networks [4]

Current approaches to classify images based on the object detection makes use of machine learning methods. To improve the performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labelled images were relatively small — on the order of tens of thousands of images (e.g., NORB [13], Caltech-101/256 [14, 15], and CIFAR-10/100 [15]).

The architecture of the network for the paper is summarised in the below figure2. It consists of the 8 layers; they are 5 convolutional and three fully connected. Here they have used the ReLU Nonlinearity as the activation function. Deep convolutional neural networks with ReLU trains faster as compared to the tanh units and can be seen in the figure 1.

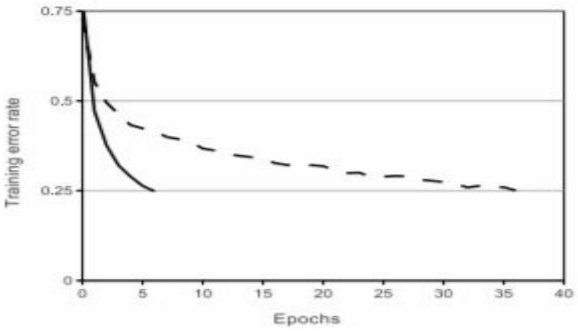


Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

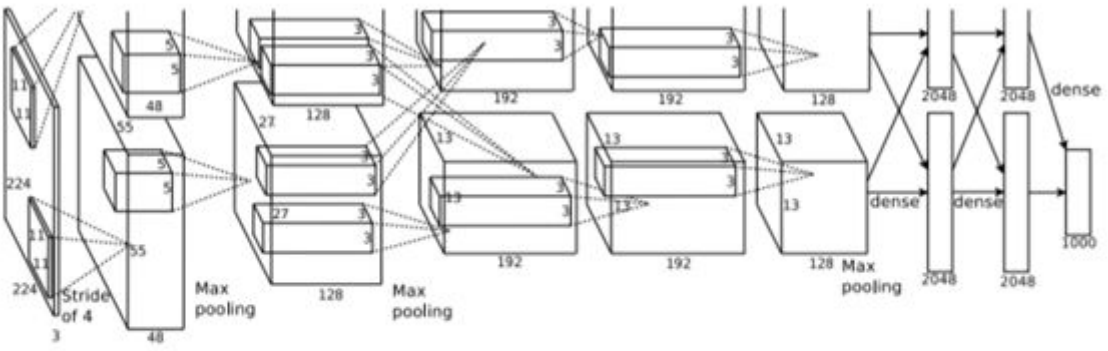


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

For the Local response normalization, since ReLU is used they have a desirable property that they do not require input normalization. Denoting by a i x,y the activity of a neuron computed by applying kernel i at position (x, y) and then applying the ReLU nonlinearity, the response-normalized activity b i x,y is given by the expression

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Pooling layers in CNN summarize the output neighbours' groups of neurons in the same kernel group. Traditionally, the neighbourhoods summarized by adjacent pooling units do not overlap (e.g., [17, 11, 4]). This is what we use throughout our network, with s = 2 and z = 3. This scheme reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, as compared with the non-overlapping scheme s = 2, z = 2, which produces output of equivalent dimensions. The disadvantage is generally we observe during training that models with overlapping pooling find it slightly more difficult to overfit.

Overall Architecture contains eight layers with weights; the first five are convolutional and the remaining three are fully connected. The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU (see Figure 2). The kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fully connected layers are connected to all neurons in the previous layer. Response-normalization layers follow the first and second convolutional layers. Max-pooling layers, of the kind described in Section 3.4, follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer.

Below table shows the results, Hence according to the research paper[12] the best performance achieved was during the ILSVRC2010 competition was 47.1% and 28.2% with an approach that averages the predictions produced from six sparse-coding models trained on different features [2], and since then the best published results are 45.7% and 25.7% with an approach that averages the predictions of two classifiers trained on Fisher Vectors (FVs) computed from two types of densely-sampled features [13].

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Convolutional Neural Network Hyper-Parameters Optimization based on Genetic Algorithms[11]

Image Classification has various applications in the research field like computer vision which further has many applications like image segmentation, object detection and localization. CNN deep architectures are basically divided into two main parts[6]. The first part, based on convolutional layers CNN, offers the ability of features extraction and input image encoding. Whereas, the second one is a fully connected neural network classifier which role is to generate a prediction model for the classification task. A CNN model is described by many hyper-parameters, specifically convolutional layers number, filter number and their respective sizes, etc.[11].

As seen from the below figure[11], the neural network functions the same as a neuron as in the human brain. Single neuron has many inputs and inputs are characterized by weights. Activation[11] function is applied where it sums all the input functions multiplied with their respective weights. To ensure that the neuron will be activated even when all entries are none, an extra input, called bias , is added. This extra input is always equal to 1 and has its own weight connection.

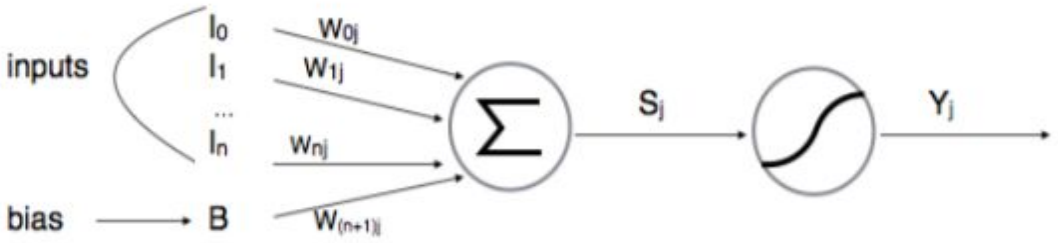


Fig. 1. Neuron Parameters.

There have been many CNN architectures developed which are an improvement over the LeNet[12] using its main concepts. Models that we quote are like AlexNet which was deeper and wider version of the LeNet, ZF Net which came as an improvement over the AlexNet by tweaking the architecture hyper-parameters, GoogleNet which when compared to the AlexNet has used the inception module that dramatically reduced the number of parameters in the network and Later came DenseNet which is the densely connected CNN has each layer directly connected to the other layer in a feed forward way. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on five highly competitive object recognition benchmark tasks.

The basic building blocks of the every neural network are Convolution, Activation function (ReLU), Pooling or Sub Sampling and Classification (Fully Connected Layer). Here they are using Genetic Algorithms with heuristic solution-search.

Genetic Algorithm[11]: GAs are heuristic solution-search or optimization methods. These techniques were originally inspired from the Darwinian principle of evolution through (genetic) selection. GAs were first proposed by John McCall [14] as a method to find best solutions to problems that were otherwise computationally intractable. McCall‟s theorem, and the related building block hypothesis, delivered a theoretical basis for the conception of effective GAs.While designing the GA algorithm following components were included:- Chromosome encoding, Fitness function form, Population size, Crossover and mutation operators and their respective rates, Evolutionary scheme to be applied and Appropriate stopping criteria .

The main aim of the research was to design an approach to be used for the image classification task. Here the author used the supervised learning algorithms for classification purposes; the input variables are labeled data (for each input from the dataset we know its class or category) and the output variable represents a category (class). The main aim of the algorithm is to learn the mapping from input to the output. Here the author has proposed the Elite CNN Model propagation(E-CNN-MP). The main structure of GA is adopted.

Algorithm 1: E-CNN-MP main algorithm

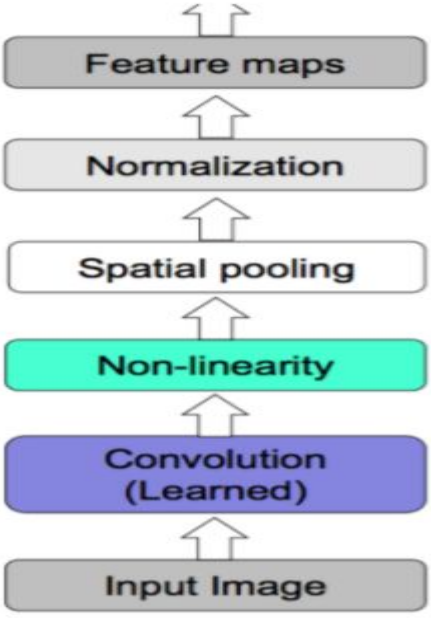
Input: D, convolutional layers number (NumConLayers), Max generations number (MaxG), Generation size N
Output: Best individual Fit*I*, Fit*CNN*
Loading images dataset D
Initialization of the training dataset (TrainingDS), the test dataset (TestDS), Best Accuracy, Fraction of elites (f_e), Fraction of crossover created children(f_c)
Randomly (Uniform distribution) create an initial population P of N chromosomes.
for g **in** 1 **to** MaxG
do
[S_1,S_2 ,..,S_N, BestAccuracy, Fit*I*, Fit*CNN*]← **FitnessCNN** (P, NumConLayers, BestAccuracy, TrainingDS, TestDS);
P'← **Recombination** (P , f_e, f_c);
P←P';
end for;
return Fit*I*;
save Fit*CNN*;

Results for the E-CNN-MP genetic algorithm with Fitness CNN function by doing batch normalization we get following result table:-

GA process	Max Accu.%	Min Accu.%	Average Accu. %	Best Network Encoding
1	85.27	0	49.81	73 49 61 96 73 18 7 8 7 16
2	85.26	0	47.73	42 75 57 48 76 2 5 11 17 16
3	89.47	0	44.55	80 80 55 70 66 14 4 2 18 2
4	86.32	0	42.11	70 70 60 77 58 17 7 3 6 9
5	87.02	0	46.85	60 73 52 76 61 11 9 3 4 2

Summary of approaches

Typical convolutional architecture consists of the below layers and doing all the below is one layer. Pooling and normalization is optional. Stack them up and train just like multi layer neural nets. Final layer is usually a fully connected neural net with output size equal to the number of classes.



1. **AlexNet**

Alexnet was developed by a group of scientists which consists of 8 layers. AlexNet consists of 5 Convolutional Layers and 3 Fully Connected Layers. Multiple Convolutional Kernels (a.k.a filters) extract interesting features in an image. In a single convolutional layer, there are usually many kernels of the same size. For example, the first Conv Layer of AlexNet contains 96 kernels of size 11x11x3. Note the width and height of the kernel are usually the same and the depth is the same as the number of channels. The first two Convolutional layers are followed by the Overlapping Max Pooling layers that we describe next. The third, fourth and fifth convolutional layers are connected directly. The fifth convolutional layer is followed by an Overlapping Max Pooling layer, the output of which goes into a series of two fully connected layers. The second fully connected layer feeds into a softmax classifier with 1000 class labels.

2. **GoogleLeNet**

The GoogLeNet architecture is very different from previous state-of-the-art architectures such as AlexNet and ZF-Net. It uses many different kinds of methods such as 1×1 convolution and global average pooling that enables it to create deeper architecture. The overall architecture is 22 layers deep. The architecture was designed to keep computational efficiency in mind. The idea behind that the architecture can be run on individual devices even with low computational resources. The architecture also contains two auxiliary classifier layers connected to the output of Inception (4a) and Inception (4d) layers.

Pros and cons:

Pros	**Cons**
Finding similarity between two comparable things and classifying them.	More hyperparameters and fine tuning is required.
Can predict out of training data set	More complex process

Relation to the Approach : We have referred to research papers based on CNN using deep neural networks. We have found different approaches are used in different papers. But we were able to relate our work with the existing approaches in a way as we are using basic architecture of the neural network that includes the Convolution, Activation function (ReLU), Pooling or Sub Sampling and Classification (Fully Connected Layer). Almost all CNN based image classification uses this basic architecture. We are also using the concept of transfer learning to train our models. While working on the project we had to read many research papers, class notes and articles on the concept of deep learning. We were able to understand the working of the CNN model with different algorithms.

Method

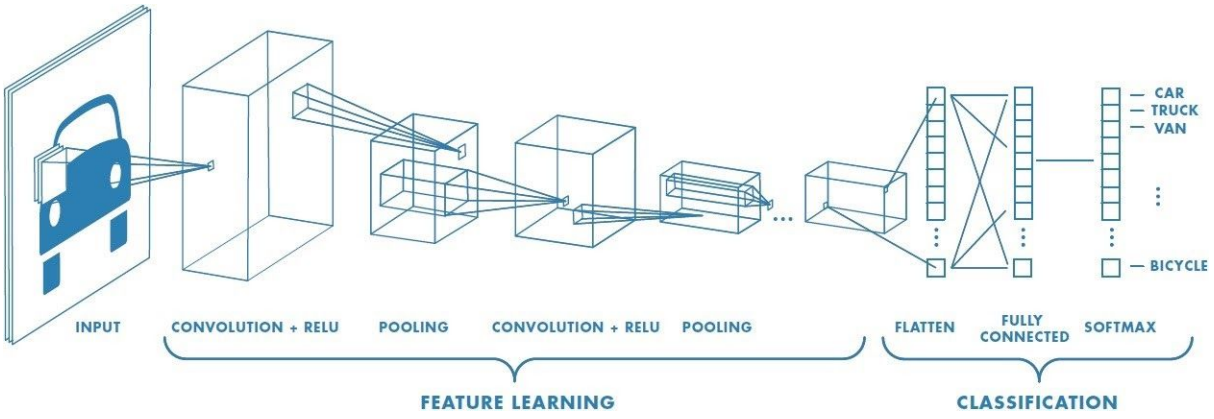
Detailed description of the Method

Installation and Configuration:- Installed all the required libraries and set up the environment. Libraries installed are like numpy, pandas, matplotlib, seaborn, tensorflow keras etc

Load and Explore the dataset:- We load and explore the data like we have around 14000 training examples and 3000 test examples. Also we calculated the proportion of observed images for each category and we found that each category has around 2500 training examples and 500 test examples. After that we also performed a few visualizations on our dataset using column chart, pie chart etc.

Image Augmentation:- Next, we have performed image augmentation using one of the libraries of Keras named ImageDataGenerator. We have done this to improve generalization properties. Firstly we rescaled our training and test data by using the rescale argument of ImageDataGenerator. After that we performed random horizontal flipping and kept some data for validation as well by using validation_split. We also zoomed the pictures according to our requirements using zoom_range.

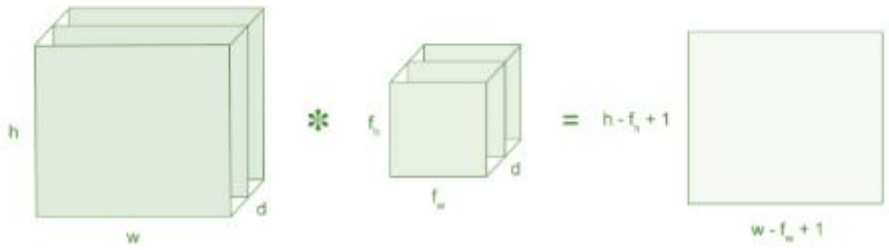
Build a Model:- We are using Convolutional Neural Network(CNN) in our project to achieve the goal. By the help of TensorFlow and Keras libraries we will build a sequential neural network which takes an input image of shape(150 * 150 *3). Computers see an input image as an array of pixels, depending on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.



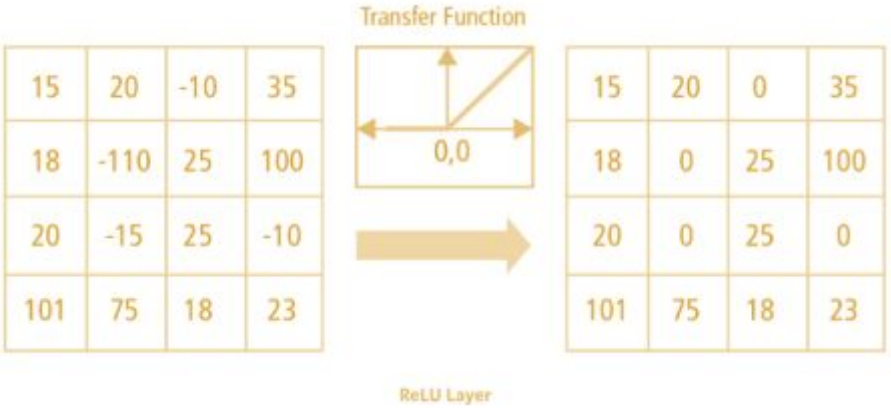
This picture is just for reference.

Convolutional Layer:- Convolution is the first layer to extract features from an input image. It preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

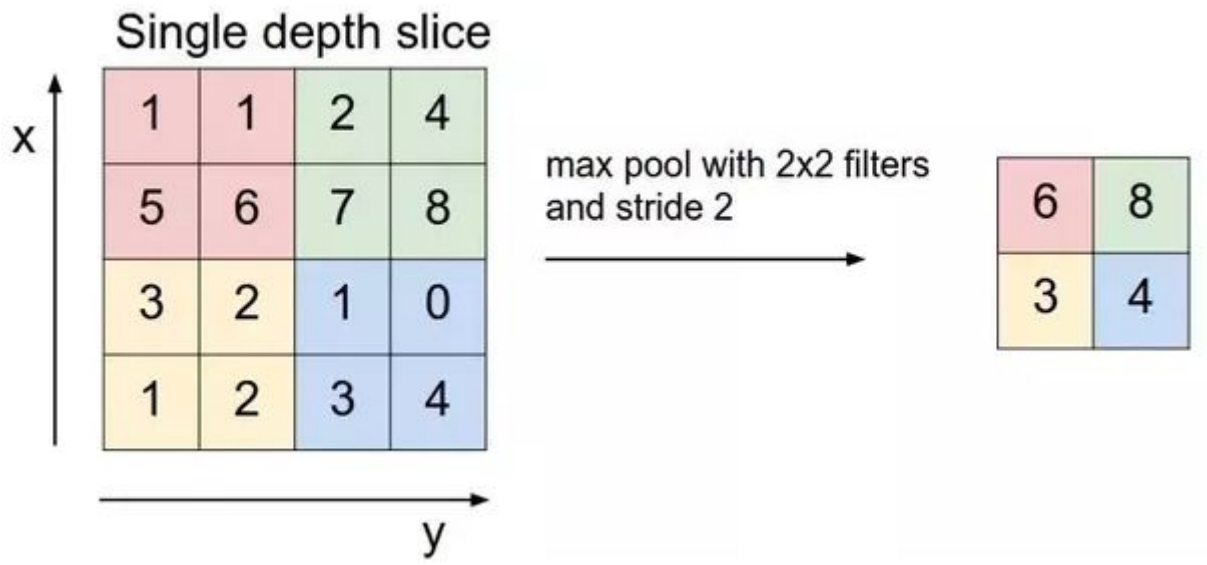
- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



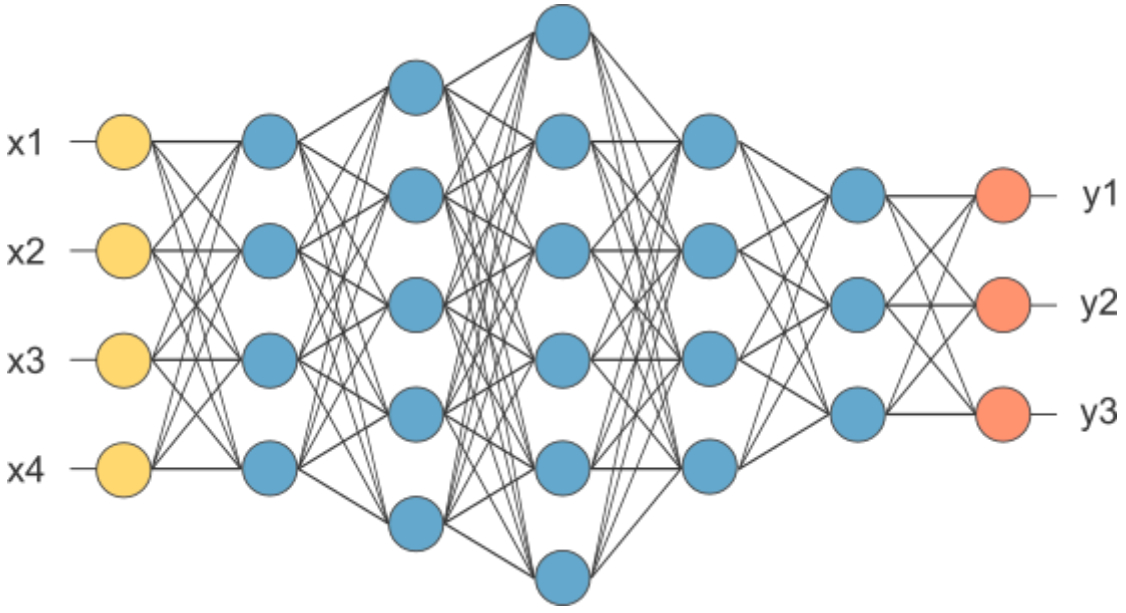
ReLU:- Rectified Linear unit an activation function whose purpose is to introduce non-linearity in the convolutional network. Given a value X, it returns max(X,0)



Pooling Layer:- It reduces the number of parameters when the image is large. In our case we are using MaxPooling, which takes the largest element from the feature map.



Fully Connected Layer:- Firstly after pooling we flattened our matrix into vector form and then fed it to the fully connected layer like a neural network. In the diagram below, the feature map matrix(X1, X2, X3 …) is converted into vector form. Now, with a fully connected layer we combined this feature to create a model. Finally we have a softmax activation function which is used to classify the outputs as buildings, glaciers, streets etc.



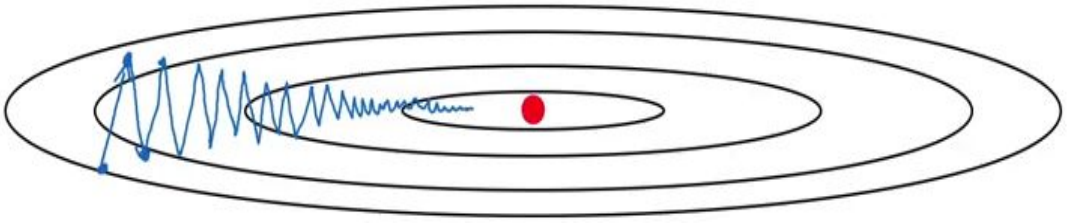
CNN Training : - Creating CNN is expensive in terms of expertise, equipment and the amount of needed data. So,we will use the concept of transfer learning in our project for image classification.In transfer learning neural network trained in two stages.First one is pretraining where the network is generally trained on a large-scale benchmark dataset representing a wide diversity of labels/categories (e.g., ImageNet). Second one is fine tuning where the pretrained network is further trained on the specific target task of interest(which is to classify different natural outlooks in our case), which may have fewer labeled examples

than the pre-training dataset. The pretraining step helps the network learn general features that can be reused on the target task.The training consists then of optimizing the network's coefficients to minimize the output classification error.

Validating Accuracy:- After compiling and training the model we will validate the accuracy by plotting the testing and validation accuracy curve as well as test and validation loss curves.

Clearly summarize your differences to existing methods:-

The paper we referenced is using heuristic search to find optimal hyperparameters of convolutional neural networks.While our approach is based on using a "stochastic gradient descent with momentum".



We want to implement a slower learning in the vertical direction and a faster learning in the horizontal direction which will help us to reach the global minimum much faster. For that we are using stochastic gradient descent with momentum.

In normal stochastic gradient descent we are updating weight and bias function in the following manner as shown in picture below

$$W = W - \text{learning rate} * dW$$

$$b = b - \text{learning rate} * db$$

In momentum instead of using dw and db independently for each instance, we take exponentially weighted average of dw and db which is shown in picture below

$$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times dW$$

$$V_{db} = \beta \times V_{db} + (1 - \beta) \times db$$

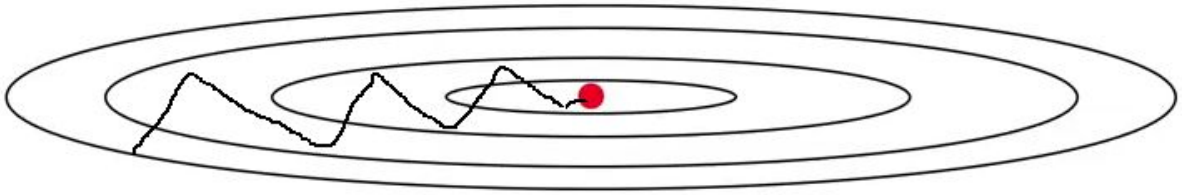
where 'β' is another hyperparameter called momentum whose values range from 0 and 1. It sets the weight between the average of previous values and the current value to calculate the new weighted average.

After the calculation of exponentially weighted average, we will update our parameters

$$W = W - \text{learning rate} * V_{dW}$$

$$b = b - \text{learning rate} * V_{db}$$

Now by using V^{dw} and V^{db} We tend to average oscillations closer to zero in the vertical direction . Although all the derivatives are pointing in the horizontal direction to the right of the horizontal direction, the horizontal direction average will still be very high. This makes it possible for our algorithm to take a more straightforward path towards local optima and dampen vertical oscillations. For this reason, the algorithm with a few iterations would end up with local optima. We can see this in the picture below



We have explained how the sgd with momentum helps us reach the global minimum faster.

Difference: Now, we are using the custom momentum rate for each epoch. For the custom momentum rate we write a function named exp_decay in which we are updating the momentum rate using the formula mentioned below:-

$$\text{momentum\_new} = m_0 \times e\text{^}(-kt)$$

where mo is the momentum value initially, -k is the decay rate and t is the current epoch.

Experiments

Explanation of experimental setup

The experimental setup consists of firstly setting up the google colab environment so that we can use data directly from kaggle without downloading it on our local system. After that we imported the libraries required and loaded our dataset and performed some visualizations. For Image classification we built six different Convolutional neural networks models. There are five main relevant methods that we are using in all five models. First is the architecture of CNN where we define the layers(like dense,flatten,conv2D,pooling,dropout etc) and activation function in layers used in the particular model. Second, we compile the model, where we mentioned the optimization, loss function and metrics we are using in a particular model. Third, we are training our model by using fit or fit_generator method on training data.Fourth we are using our test data to measure the accuracy of the model on the dataset. Fifth, we are plotting two graphs, first is showing the difference between training and validation accuracy. Second is showing the difference between training and validation loss of the model.

Test results of the proposed method

The accuracy of the proposed model is 67 percent which is not very good.But we have successfully implemented the proposed method. Below is the screenshot attached.

```
Epoch 15/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8989 - accuracy: 0.6516 - val_loss: 0.9159 - val_accuracy: 0.6381
Epoch 16/25
54/54 [==============================] - 6s 116ms/step - loss: 0.8826 - accuracy: 0.6601 - val_loss: 0.8909 - val_accuracy: 0.6543
Epoch 17/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8676 - accuracy: 0.6681 - val_loss: 0.8872 - val_accuracy: 0.6537
Epoch 18/25
54/54 [==============================] - 6s 116ms/step - loss: 0.8649 - accuracy: 0.6668 - val_loss: 0.8928 - val_accuracy: 0.6540
Epoch 19/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8498 - accuracy: 0.6711 - val_loss: 0.9239 - val_accuracy: 0.6349
Epoch 20/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8425 - accuracy: 0.6806 - val_loss: 0.8927 - val_accuracy: 0.6586
Epoch 21/25
54/54 [==============================] - 6s 114ms/step - loss: 0.8299 - accuracy: 0.6829 - val_loss: 0.8617 - val_accuracy: 0.6637
Epoch 22/25
54/54 [==============================] - 6s 114ms/step - loss: 0.8241 - accuracy: 0.6867 - val_loss: 0.8416 - val_accuracy: 0.6814
Epoch 23/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8181 - accuracy: 0.6852 - val_loss: 0.8291 - val_accuracy: 0.6882
Epoch 24/25
54/54 [==============================] - 6s 115ms/step - loss: 0.8054 - accuracy: 0.6943 - val_loss: 0.8470 - val_accuracy: 0.6794
Epoch 25/25
54/54 [==============================] - 6s 114ms/step - loss: 0.8094 - accuracy: 0.6908 - val_loss: 0.8445 - val_accuracy: 0.6783
```

```
test_loss = model_exp.evaluate(test_images, test_labels)
```

```
94/94 [==============================] - 1s 8ms/step - loss: 0.8460 - accuracy: 0.6787
```

The screenshot below shows the result of the VGG19 model with dropout and LeakyRelu and without image augmentation. This is our best result out of all six models.

```
21/21 [==============================] - 0s 14ms/step - loss: 0.2301 - accuracy: 0.9152 - val_loss: 0.3596 - val_accuracy: 0.8829
Epoch 15/25
21/21 [==============================] - 0s 13ms/step - loss: 0.2158 - accuracy: 0.9205 - val_loss: 0.3708 - val_accuracy: 0.8792
Epoch 16/25
21/21 [==============================] - 0s 13ms/step - loss: 0.2115 - accuracy: 0.9226 - val_loss: 0.3849 - val_accuracy: 0.8740
Epoch 17/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1940 - accuracy: 0.9276 - val_loss: 0.4135 - val_accuracy: 0.8723
Epoch 18/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1823 - accuracy: 0.9321 - val_loss: 0.3929 - val_accuracy: 0.8775
Epoch 19/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1733 - accuracy: 0.9376 - val_loss: 0.3931 - val_accuracy: 0.8809
Epoch 20/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1569 - accuracy: 0.9450 - val_loss: 0.3972 - val_accuracy: 0.8846
Epoch 21/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1514 - accuracy: 0.9440 - val_loss: 0.4295 - val_accuracy: 0.8735
Epoch 22/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1512 - accuracy: 0.9448 - val_loss: 0.3896 - val_accuracy: 0.8846
Epoch 23/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1409 - accuracy: 0.9495 - val_loss: 0.4432 - val_accuracy: 0.8692
Epoch 24/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1315 - accuracy: 0.9523 - val_loss: 0.4487 - val_accuracy: 0.8780
Epoch 25/25
21/21 [==============================] - 0s 13ms/step - loss: 0.1259 - accuracy: 0.9551 - val_loss: 0.4418 - val_accuracy: 0.8829
```

```
test_loss = model4.evaluate(test_features, test_labels)
```

```
94/94 [==============================] - 0s 2ms/step - loss: 0.4401 - accuracy: 0.8740
```

This table shows the accuracy of each model along with which model uses the concept of transfer learning and Image augmentation.

Model	**Image Augmentation**	**Transfer Learning**	**Accuracy**
Model 1	No	No	67.87
Model 2	No	No	72.73
Model 3	No	No	79.83
Model 4	No	Yes	87.03
Model 5	No	Yes	87.40
Model 6	Yes	No	81.30

Deep analysis/discussion about the results

1 - Models without Image Augmentation

1.1 Models without Transfer Learning

Model 1

We build this CNN using few convolutional, maxpooling, flatten, dense and dropout layer. But one thing we did differently is using Stochastic gradient descent optimization with the custom momentum rate for which we used the formula explained in the differences section. The model accuracy is not very good which is 67 percent. However, we implemented a method different from the existing ones.

Model 2

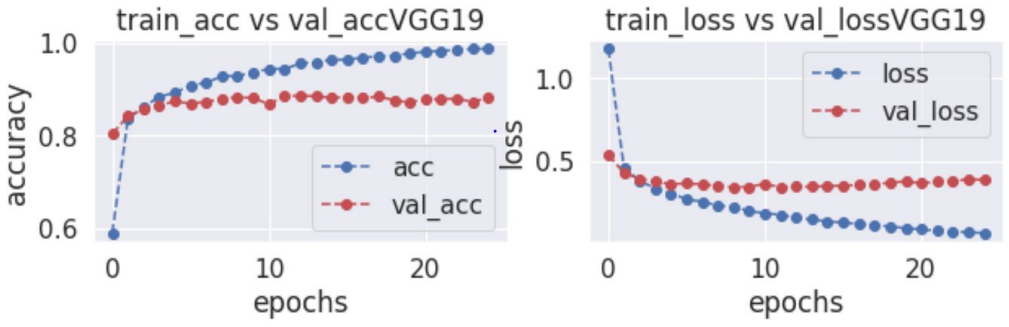
Firstly, we load and scale the data without using image augmentation. We then build a simple convolutional neural network which contains few convolution, maxpooling, flatten and dense layer. We used relu as an activation function for all the hidden layers and softmax output layer. We achieved an accuracy around 72.73 percent which is not bad.

Model 3

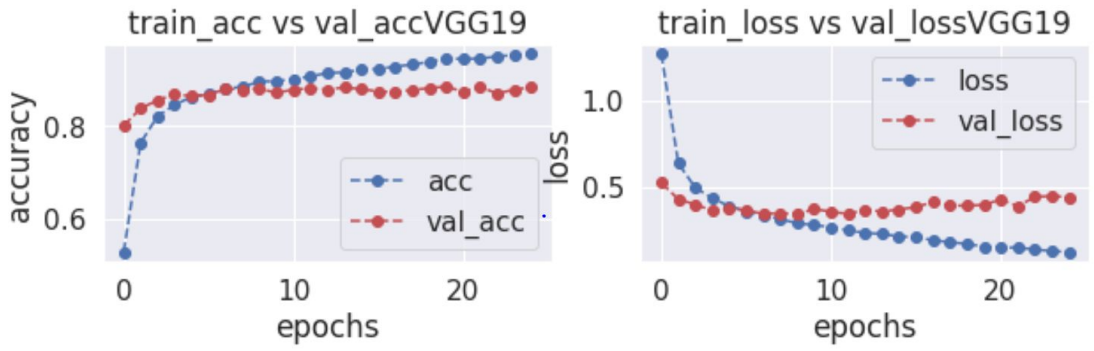
Secondly, we build another Concolutional neural network with an extra layer of convolution, maxpooling and also added dropout layer for regularization. We used LeakyRelu this time as an activation function for dense layers.We achieved an accuracy of 79.83 percent, which is a good jump from the previous model.

1.2 VGG19 Model

After that, we used the concept of transfer learning and imported the VGG19 model using "imagenet" as weights. We freezed the upper layer and built a simple neural network for the upper layers and used Relu as an activation function. We achieved the accuracy of 87.03 percent which is much greater than the models without transfer learning. After that we have added a few more dense layers and added dropout for regularization, as well as we used Leaky Relu as an activation function. We see from the graph that the difference between training accuracy and validation accuracy are lesser than the previous model.Also, we achieved the accuracy of 87.40.



VGG19 with Relu



VGG19 with more dense layer and Leaky Relu

2 - Model with Image Augmentation

2.1 Model without Transfer Learning

Model 4

I have used the same model architecture as I used for the model 1. Relu as an activation function for hidden layers and softmax for output layers. After performing the Image augmentation method we got a huge increase accuracy almost more than 9 percent, which is very good.

Amount of effort team has made

We have used CNN with some optimization techniques to classify the images in the data set. We read many research papers and different architectures to understand the working of CNN and how to use these techniques to get better results. We have then set up the environment to run our model, Model implementation, try some different and better approach and validation of the results.

Task	Apnav	Ashesh	Pooja	Samarth	Sapna
Project Survey & Topic selection	✓	✓	✓	✓	✓
Dataset Selection	✓	✓	✓		
Literature Survey		✓	✓		✓
Analysis of Data	✓	✓		✓	✓
Planning		✓		✓	✓

Project Proposal	✓	✓	✓	✓	✓
Data Pre-processing		✓	✓	✓	✓
Libraries	✓	✓			✓
Model selection	✓	✓	✓	✓	✓
Mid Progress Report	✓	✓	✓	✓	✓
Model Training	✓	✓	✓		✓
Project Presentation	✓	✓	✓	✓	✓
Final Report	✓	✓	✓	✓	✓

Reflections

List of summary of comments by instructor

1) I see you have worked on making some difference. Can you clarify how different your optimization is from the momentum optimization and other variants?

   We are using the custom momentum rate for each epoch instead of fixed momentum rate. We are using the following formula for custom momentum rate

$$\text{momentum\_new} = m_0 \times e^{\wedge}(-kt)$$

2) Format

   We had submitted in document format pdf. This time we will submit the report in the proper mentioned format.

References & Citations

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "Image{N}et classification with deep convolutional neural networks,"in Proceedings of the Advances in neural information processing systems, pp. 1097-1105, 2012.

[2] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3128-3137, 2015.

[3] T. Kokul, C. Fookes, S. Sridharan, A. Ramanan, and U. Pinidiyaarachchi, "Gate connected convolutional neural network for object tracking," in IEEE International Conference on Image Processing (ICIP), pp. 2602-2606, 2017.

[4] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3{D} {T}raffic {S}cene {U}nderstanding {F}rom{M}ovable {P}latforms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, pp. 1012-1025, 2014.

[5] N. Rasheed, S. A. Khan, and A. Khalid, "Tracking and {A}bnormal {B}ehavior {D}etection in {V}ideo{S}urveillance {U}sing {O}ptical {F}low and {N}eural {N}etworks," in Proceedings of the AdvancedInformation Networking and Applications Workshops pp. 61-66, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprintarXiv:1512.03385, 2015.

[7] N. Jmour, S. Zayen and A. Abdelkrim, "Convolutional neural networks for image classification," 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Hammamet, 2018, pp. 397-402, doi: 10.1109/ASET.2018.8379889.

[8]. Neha Sharma,Vibhor Jain,Anju Mishra,“An Analysis Of Convolutional Neural Networks For Image Classification”,Procedia Computer Science,Volume132,2018(http://www.sciencedirect.com/science/article/pii/S1877050918309335).

[9]. Jaswal, Deepika & Vishvanathan, Sowmya & Kp, Soman. (2014). Image Classification Using Convolutional Neural Networks. International Journal of Scientific and Engineering Research. 5. 1661-1668. 10.14299/ijser.2014.06.002.

[10]. Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, et al., "Large-scale image classification: fast feature extraction and svm training," in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on , pp. 1689-1696,2011.

[11] Sehla Loussaief1 , Afef Abdelkrim2, “Convolutional Neural Network Hyper-Parameters Optimization based on Genetic Algorithms” in (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 10, 2018.

[12] http://d2l.ai/chapter_convolutional-neural-networks/lenet.html

[13] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–97. IEEE, 2004

[14] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding, 106(1):59–70, 2007.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, Department of Computer Science, University of Toronto, 2009.

[16] Chandrarathne, Gayani & Thanikasalam, Kokul & Pinidiyaarachchi, Amalka. (2019). A Comprehensive Study on Deep Image Classification with Small Datasets.

Conclusion

Concluding remarks

Our Solution using CNN has shown efficient prediction results. We have also seen techniques like image augmentation, max-pooling, dropout and the use of more advanced activation functions (Leaky ReLu) greatly improving the prediction results. This is a challenging task when looking at the images. This may be due to the choice of 150X150 pixels or that 72X72 or 16X16 pixels are not well suited for these situations. The learning capacity of CNN is significantly improved over the years by exploiting depth and other structural modifications. The exploitation of different innovative ideas in CNN architectural design has changed the direction of research, especially in image processing and CV. Good performance of CNN on a grid-like topological data presents it as a powerful representational model for images. Architectural design of CNN is a promising research field and in future, it is likely to be one of the most widely used AI techniques.

Thoughts about the project

We worked on the Image classification using CNN. We read about different approaches to classify the images and were able to develop good knowledge of CNN. And different techniques to solve the image classification problems. We were able to learn deep neural network architectures in CNN and develop an in-depth understanding of the CNN architecture. While working on the project we learned about the CNN, neural networks and optimization techniques to achieve the high level accuracy. We got good hands on the python libraries like Tensorflow and Keras.

Challenges and how to overcome them

Our main challenge was to identify all the images of different categories correctly and build a model that predicts any new image with the correct label from the given categories. Also,if the images are taken in different lighting conditions, at different angles or varying colors in color images, Image classifiers might get confused with other similar pictures of different categories.

We used the concept of Data Augmentation which takes the approach of generating additional images from our existing dataset by augmenting them using random transformations that yield similar looking images. It will help models to generalize better and recognize objects in different angles.

GitHub Link

https://github.com/poojaelkal/MachineLearning_Project