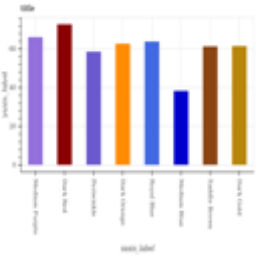
Chart Image Classification

Revant Teotia

July 2020

Analyzing the dataset

1. Dataset contains 1000 images
2. Images belong to 5 classes (types of chart) : "Line","Dot Line","Horizontal Bar","Vertical Bar", and "Pie"
   (a) Each class has 200 examples
3. Each image is 128x128 RGB image



Example image of a chart

Splitting Dataset into Training and Validation

Steps involved in splitting the dataset

1. Randomly shuffled the dataset
2. For each class (i.e. each type of chart), used 80% for training and 20% for validation
   (a) Training dataset has 800 examples : 160 images for each class
   (b) Validation dataset has 200 examples : 40 images for each class

Training a two layer CNN to classify chart images

Using "Cross Entropy Loss" as loss function and 'Adam' optimizer

Data Preprocessing before training

Steps involved in preprocessing :

1. Rescaled pixels values from 0-255 to 0-1
2. Batched the data into batches of 32 images

Model Architecture

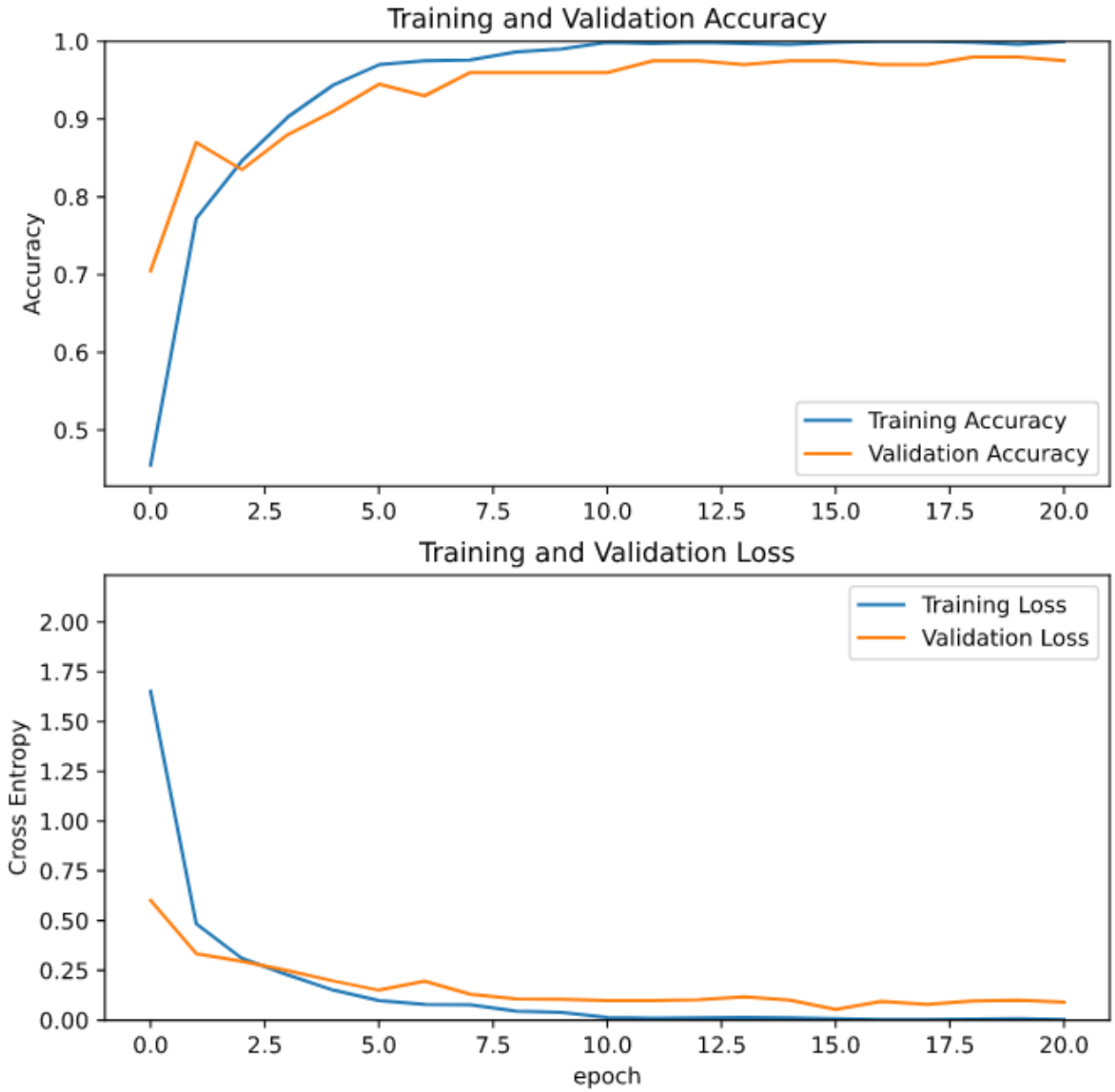
Image (128x128)
↓
convolution Layer (variable filters, kernel size = (3,3) and stride = (1,1), relu activation)
↓
Maxpooling Layer (pool size = (2,2) and stride = (2,2))
↓
convolution Layer (variable filters, kernel size = (3,3) and stride = (1,1), relu activation)
↓
Maxpooling Layer (pool size = (2,2) and stride = (2,2))
↓
Flattening Layer
↓
One or more Dense Layers with Droupouts (with relu activation)
↓
Output Layer (10 units with softmax activation)

Observations noted after training

After trying out different number of Filters for convolution layers, number of units in dense layers and drouput rates, following observations are noted :

1. Without using dropout :
   (a) Observed **overfitting** : **validation accuracy : 98-99% while training accuracy reaches 100%**
   (b) Overfitting increased when number of filters in convolution layers were increased : i.e. increasing model complexity increased the overfitting
2. Using dropout : Reduced overfitting and slightly improved accuray on validation set.
   (a) But did not have a large impact on validation accuracy.

Progress of loss and accuracy during training of model with : 64 filters in each convolution layer, a dense layer with 128 units after convolution layers with dropout(of 50%) :



More detailed observations and experiments with code can be found here

Transfer Learning : using VGG16 with weights trained on imagenet dataset

Steps involved :

1. Preprocessing the images.
2. Freezing the weights of all convolution blocks of pre-trained VGG16 by making them untrainable
3. Removing fully connected layers (on top of the convolution blocks) of the pre-trained VGG16
4. Adding fully connected layers on top of freezed convolution blocks and training them to fit our dataset

Data Preprocessing before training

Steps involved in preprocessing for VGG16 input:

1. Resizing images to 224x224
2. Converting images from RGB to BGR
3. Zero-centering each color channel with respect to the ImageNet dataset, without scaling

Model Architecture

Image (224x224)
↓
Preprocessing (RGB to BGR and zero-centering wrt imagenet)
↓
VGG16 Convolution layers (with untrainable pre-trained weights)
↓
Global Average Pooling Layer
↓
Dense Layer (with 64 units and relu activation)
↓
Dropout Layer (50% dropout rate)
↓
Output Layer (10 units with softmax activation)

Observations noted after training

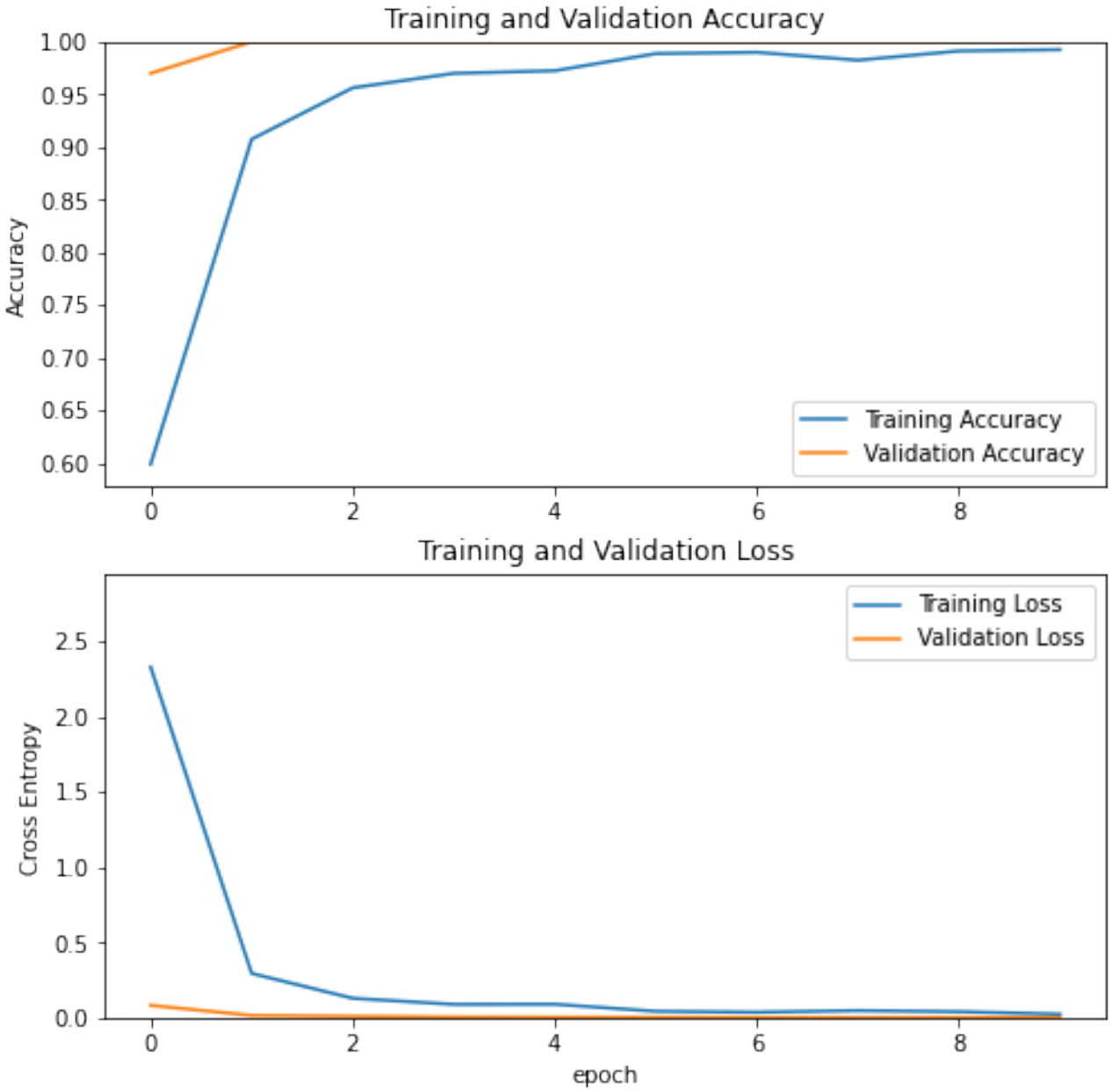
Based on loss and accuracy plot on next page.

1. Performs Really well : reaching 100% validation accuracy after training for only 2-3 epochs
2. No overfitting

Why does transfer learning work really well ?

Because convolution layers of a network trained on large datasets (like imagenet with more than 14 million images) are already learned to extract features from the images, and when it is trained to classify using a new dataset it only needs to classify images based on the extracted features from from the images. While an untrained network has to learn feature extraction also which is not very effective when working with small datasets.

Progress of loss and accuracy during training of the above model:



More detailed observations and experiments with code can be found here

Code used :

(also containing more details of experiments done) :

- For data analysis and train-validation split
- For classification using 2 layer CNN
- For classification using transfer learning