
# *Image Classification using Neural Networks*

Submitted by:

Utsav Rastogi

# ACKNOWLEDGMENT

This project was my first attempt at attempting a computer vision problem. This required intensive learning and thorough research on how to build models and the preliminary steps as I was not informed at all about this subfield of Machine Learning. I'm highly fortunate to have got this experiential opportunity along with the completion of my project work. I am also grateful to Flip robo Technologies for assigning this project to me.

Multiple references were used such as:

ImageNet, Analytics Vidhya, Medium, GitHub and Stack Overflow

# INTRODUCTION

- Business Problem Framing

  Image classification of several apparel of different categories from amazon.com and flipkart.com. The use of such algorithms can largely reduce the manual labour and automate the supply chain where categorizing on a large scale can be a tedious process

- Conceptual Background of the Domain Problem

  Image classification is an essential part in the modern supply chain. It can be used to enhance the capabilities of modern devices with exceptional accuracy and increasing their reliability in day-to-day tasks. Several examples where Image recognition is used today: Identifying symbols and activities in a driverless car, snapchat filters, movies where heavy CGI is being used, Video games, identifying land use via Geospatial statistics using satellite imagery, mapping the ocean floor, VR and Healthcare. The possibilities of computer vision are endless and are bound to expand exponentially in the coming decade.

  Segmentation is a necessary step for a multitude of large-scale retail and whole sale business such as Walmart, Target, Bigbazaar etc. It's not just limited to apparels but with a large enough dataset, machines can be trained to identify a large variety of objects, reducing the use of manual labour and increasing the efficiency of sorting.

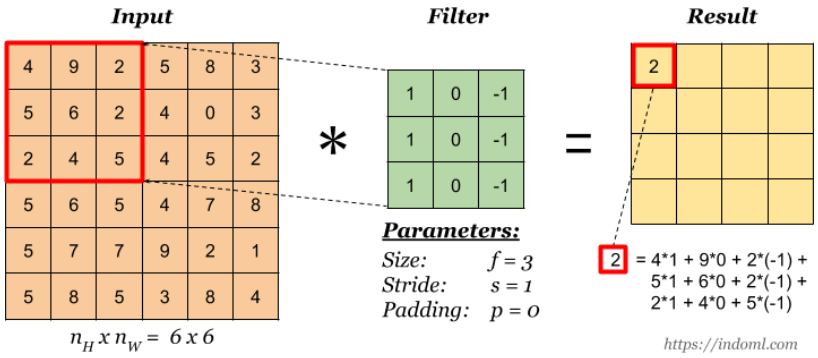
- Review of Literature

  ## CNN

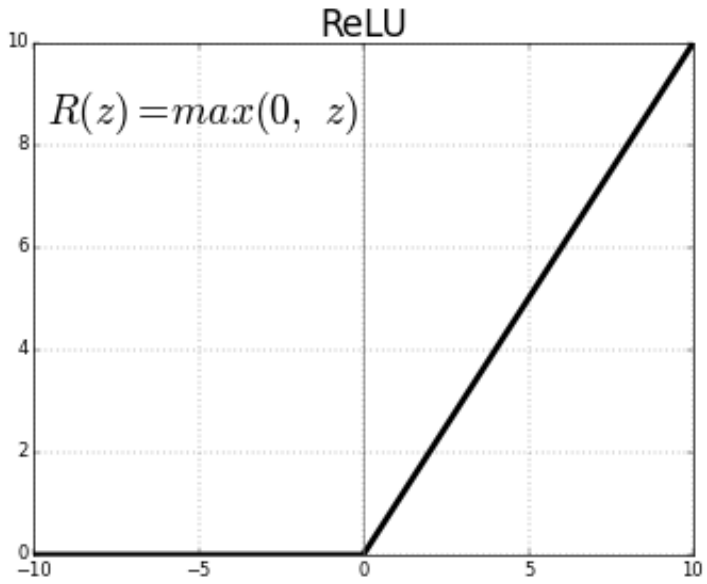
  Architecture of Convolutional neural network Convolutional neural network is a type of artificial neural network that uses multiple perceptron that analyse image inputs and have learnable weights and bases to several parts of images and able to segregate each

other. One advantage of using Convolutional Neural Network is it leverages the use of local spatial coherence in the input images, which allow them to have fewer weights as some parameters are shared. This process is clearly efficient in terms of memory and complexity. The basic building blocks of convolutional neural network are as follows:

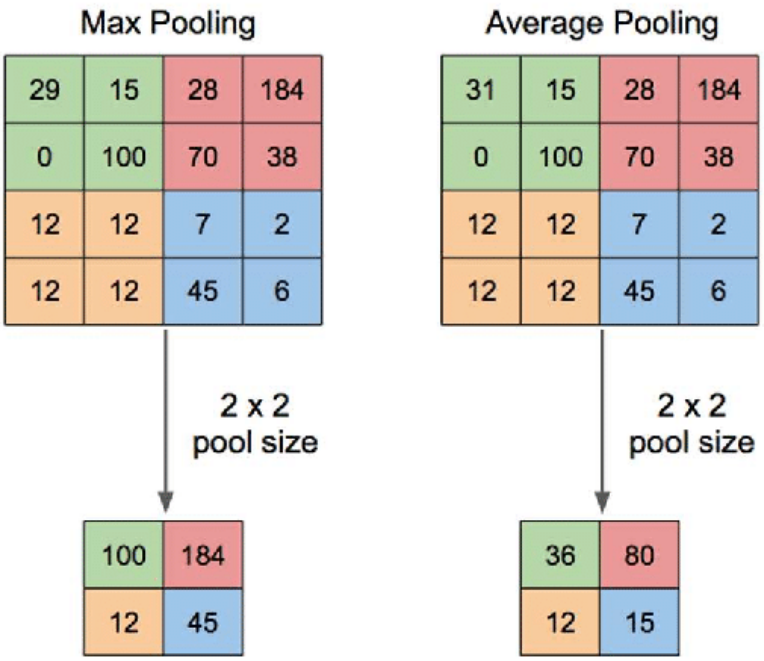
a. Convolution Layer – In convolutional layer, a matrix named kernel is passed over the input matrix to create a feature map for the next layer. We execute a mathematical operation called convolution by sliding the Kernel matrix over the input matrix. At every location, an element wise matrix multiplication is performed and sums the result onto the feature map. Convolution is a specialized kind of linear operation which is widely used in variety of domains including image processing, statistics, physics. Convolution can be applied over more than 1 axis. If we have a 2-Dimensional image input, I, and a 2-Dimensional kernel filter, K, the convoluted image is calculated as follows:



b. Non-linear activation functions (ReLU) – Activation function is a node that comes after convolutional layer and the activation function is the nonlinear transformation that we do over the input signal. The rectified linear unit activation function (ReLU) is a piecewise linear function that will output the input if is positive, otherwise it will output zero.

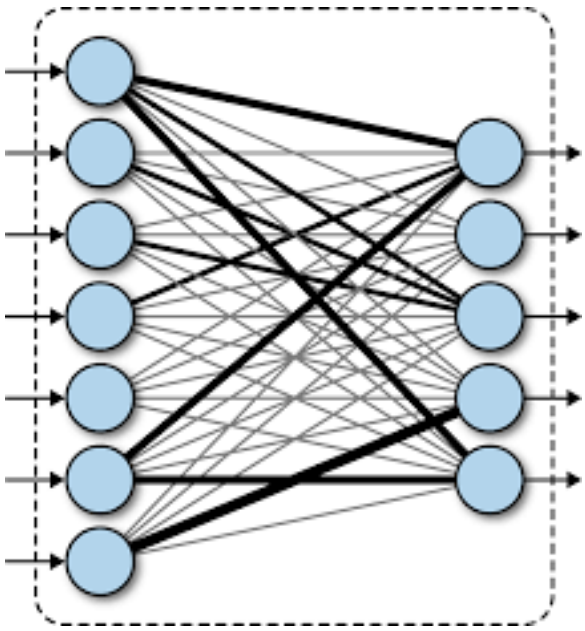



c. Pooling Layer – The drawback of the feature map output of convolutional layer is that it records the precise position of features in the input. This means during cropping, rotation or any other minor changes to the input image will completely results in a different feature map. To counter this problem, we approach down sampling of convolutional layers. Down sampling can be achieved by applying a pooling layer after nonlinearity layer. Pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.



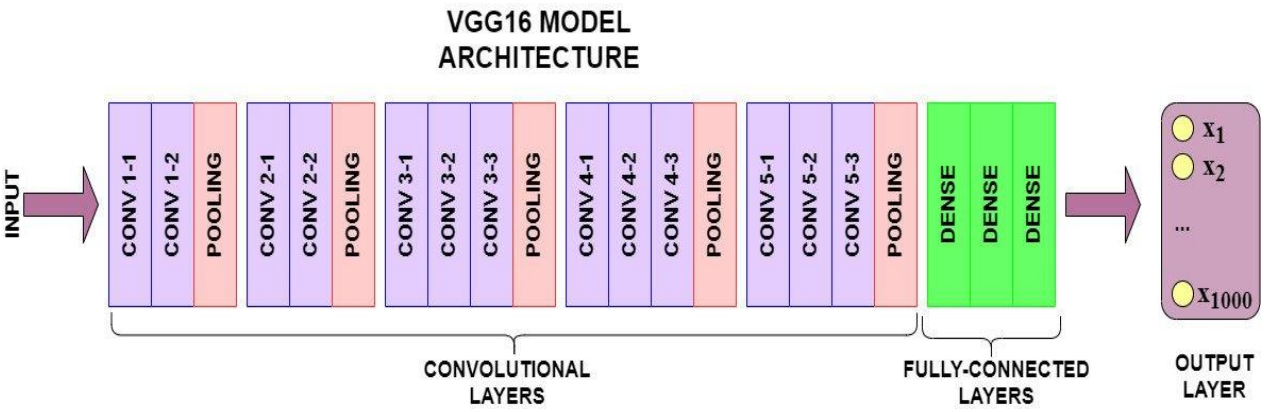

d. Fully Connected Layer - At the end of a convolutional neural network, the output of the last

Pooling Layer acts as input to the Fully Connected Layer. There can be one or more of these layers. Fully connected means that every node in the first layer is connected to every node in the second layer.



ImageNet is a research project to develop a large database of images with annotations e.g. images and their labels. Pretrained models like InceptionV1, Inception V2, VGG-16 and VGG-19 are already trained on ImageNet which comprises of disparate categories of images. These models are built from scratch and trained by using high GPUs over millions of images consisting of thousands of image categories. As the model is trained on huge dataset, it has learned a good representation of low-level features like spatial, edges, rotation, lighting, shapes and these features can be shared across to enable the knowledge transfer and act as a feature extractor for new images in different computer vision problems. These new images might be of completely different categories from the source dataset, but the pretrained model should still be able to extract relevant features from these images based on the principles of transfer learning. In this paper we will unleash the power of transfer learning by using pretrained model - VGG-16 as an effective feature extractor to classify dog vs cat even with fewer training images.

## Transfer Learning using VGG16




VGG-16 model architecture – 13 convolutional layers and 2 Fully connected layers and 1 SoftMax classifier VGG-16 - Karen Simonyan and Andrew Zisserman introduced VGG-16 architecture in 2014 in their paper Very Deep Convolutional Network for Large Scale Image

Recognition. Karen and Andrew created a 16-layer network comprised of convolutional and fully connected layers. Using only 3×3 convolutional layers stacked on top of each other for simplicity.

The precise structure of the VGG-16 network shown in Figure. 7. is as follows:

• The first and second convolutional layers are comprised of 64 feature kernel filters and size of the filter is 3×3. As input image (RGB image with depth 3) passed into first and second convolutional layer, dimensions changes to 224x224x64. Then the resulting output is passed to max pooling layer with a stride of 2.

• The third and fourth convolutional layers are of 124 feature kernel filters and size of filter is 3×3. These two layers are followed by a max pooling layer with stride 2 and the resulting output will be reduced to 56x56x128.

• The fifth, sixth and seventh layers are convolutional layers with kernel size 3×3. All three use 256 feature maps. These layers are followed by a max pooling layer with stride 2.

• Eighth to thirteen are two sets of convolutional layers with kernel size 3×3. All these set of convolutional layers have 512 kernel filters. These layers are followed by max pooling layer with stride of 1.

• Fourteen and fifteen layers are fully connected hidden layers of 4096 units followed by a SoftMax output layer (Sixteenth layer) of 1000 units.

# Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

- Data Sources and their formats

  Images are scraped for 3 different categories and stored under respective folder named: Sarees, Jeans(men) and trousers(men). This was done for both training and test set. Both of them are balanced with equal number of images for each category.

  1. Training set: Amazon.in (total: 742 images)
  2. Test set: Flipakrt.com (total: 356 images)

- Hardware and Software Requirements and Tools Used

  Hardware:

  • RAM: 16 GB

  • CPU: Intel® Core™ i7-6500U CPU @ 3.10GHz

  • GPU: NVIDIA- GeForce GTX 960M

  Software:

  • Programming language: Python

  • Distribution: Anaconda Navigator

  • Browser based language shell: Jupyter Notebook

  • Libraries: Pandas, NumPy, matplotlib, keras, TensorFlow

# Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

    1. VGG16
    2. VGG19
    3. Resnet

- Run and evaluate selected models

In [6]:
```
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
vgg19 = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

...

- "imagenet" weights have been used.
- false, means last layer is being eliminated
- not training the existing layers below, so specified False

In [7]:
```
for layer in vgg16.layers:
    layer.trainable = False

for layer in vgg19.layers:
    layer.trainable = False

for layer in resnet.layers:
    layer.trainable = False
```

In [8]:
```
## Flatten the output - converting the data into a 1d array for input to next layer

x1 = Flatten()(vgg16.output)

x2 = Flatten()(vgg19.output)

x3 = Flatten()(resnet.output)
```

In [33]:
```
#checking the folder
folder = glob(r"C:\Users\utsav\Desktop\datatrained\internship\Assignment 13\img_data\train\*")
```

In [35]:
```
len(folder)
```

Out[35]: 3

In [36]:
```
#training prediction

pred1 = Dense(len(folder), activation='softmax')(x1)
pred2 = Dense(len(folder), activation='softmax')(x2)
pred3 = Dense(len(folder), activation='softmax')(x3)
```

In [37]:
```
##final model
#creating the object named model
model16 = Model(inputs=vgg16.input, outputs=pred1)
model19 = Model(inputs=vgg19.input, outputs=pred2)
modelres = Model(inputs=resnet.input, outputs=pred3)
```

**Model structure**

In [38]:
```
print("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-VGG 16-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-")
print(model16.summary())

print("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-VGG 19-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-")
print(model19.summary())

print("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-RESNET-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-")
print(modelres.summary())
```

```
#Cost and optimization method
model16.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
model19.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
modelres.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
#data augmentation
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)

training_set = train_datagen.flow_from_directory(r'C:\Users\utsav\Desktop\datatrained\internship\Assignment 13\img_data\
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 740 images belonging to 3 classes.
```

```
#data augmentation
test_datagen = ImageDataGenerator(rescale = 1./255)

test_set = test_datagen.flow_from_directory(r'C:\Users\utsav\Desktop\datatrained\internship\Assignment 13\img_data\test
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
#fitting the model

model16_fit = model16.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

- - -

```
model19_fit = model19.fit(
  training_set,
  validation_data=test_set,
  epochs=14,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set))
```

- - -

```
modelres_fit = modelres.fit(
  training_set,
  validation_data=test_set,
  epochs=12,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set))
```

- - -

- Key Metrics for success in solving problem under consideration

xxxxxxxxxxxxxxxxxxxxxxxVGG16xxxxxxxxxxxxxxxxxxx

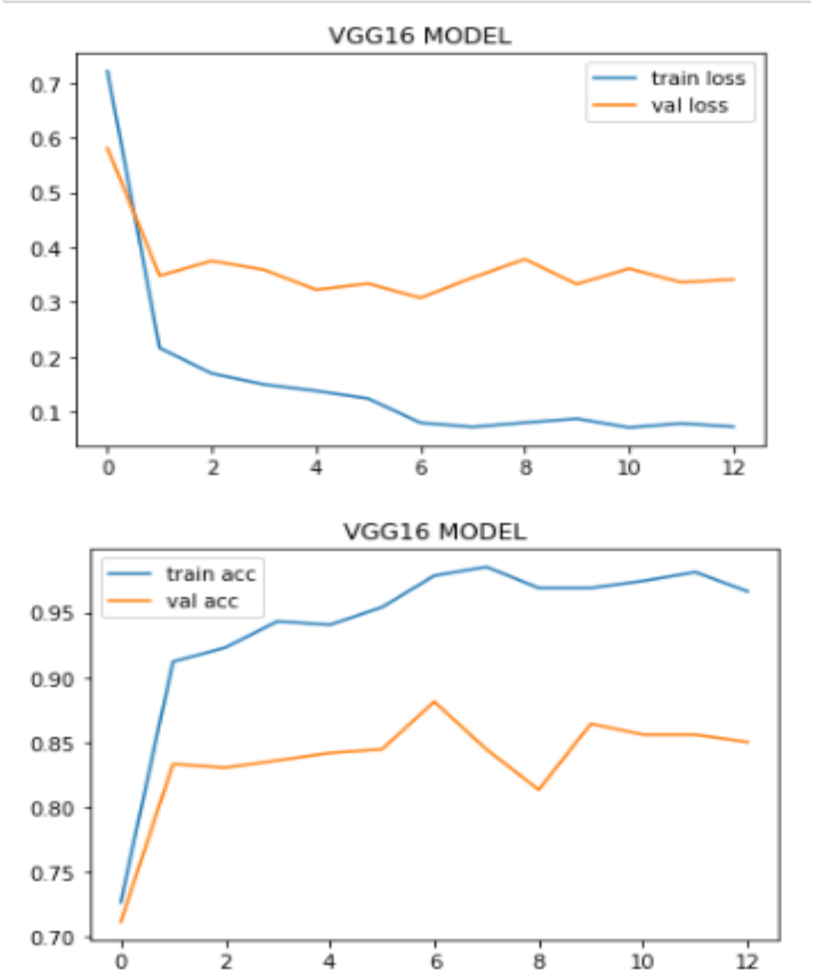
| | loss | accuracy | val_loss | val_accuracy |
|---|---|---|---|---|
| 0 | 0.721584 | 0.727027 | 0.581177 | 0.711864 |
| 1 | 0.216846 | 0.912162 | 0.348703 | 0.833333 |
| 2 | 0.170201 | 0.922973 | 0.375484 | 0.830508 |
| 3 | 0.150432 | 0.943243 | 0.359378 | 0.836158 |
| 4 | 0.138532 | 0.940541 | 0.323054 | 0.841808 |
| 5 | 0.124534 | 0.954054 | 0.334133 | 0.844633 |
| 6 | 0.080384 | 0.978378 | 0.308177 | 0.881356 |
| 7 | 0.072674 | 0.985135 | 0.344923 | 0.844633 |
| 8 | 0.080672 | 0.968919 | 0.378730 | 0.813559 |
| 9 | 0.087950 | 0.968919 | 0.333160 | 0.864407 |
| 10 | 0.071720 | 0.974324 | 0.361623 | 0.855932 |
| 11 | 0.079264 | 0.981081 | 0.336446 | 0.855932 |
| 12 | 0.073109 | 0.966216 | 0.341709 | 0.850282 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxVGG19xxxxxxxxxxxx

| | loss | accuracy | val_loss | val_accuracy |
|---|---|---|---|---|
| 0 | 0.684355 | 0.754054 | 0.593821 | 0.711864 |
| 1 | 0.259729 | 0.900000 | 0.432190 | 0.762712 |
| 2 | 0.223339 | 0.900000 | 0.344851 | 0.810734 |
| 3 | 0.197906 | 0.916216 | 0.523718 | 0.754237 |
| 4 | 0.180691 | 0.916216 | 0.659272 | 0.754237 |
| 5 | 0.175456 | 0.932432 | 0.316897 | 0.847458 |
| 6 | 0.113237 | 0.960811 | 0.315191 | 0.841808 |
| 7 | 0.099048 | 0.970270 | 0.333315 | 0.838983 |
| 8 | 0.093219 | 0.975676 | 0.375162 | 0.816384 |
| 9 | 0.092811 | 0.970270 | 0.369587 | 0.813559 |
| 10 | 0.084052 | 0.971622 | 0.328204 | 0.838983 |

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx RESNET xxxxxx

| | loss | accuracy | val_loss | val_accuracy |
|---|---|---|---|---|
| 0 | 4.254151 | 0.474324 | 0.937988 | 0.658192 |
| 1 | 0.705145 | 0.682432 | 0.618894 | 0.720339 |
| 2 | 0.493293 | 0.758108 | 0.813201 | 0.677966 |
| 3 | 0.487345 | 0.766216 | 0.756663 | 0.675141 |
| 4 | 0.575057 | 0.758108 | 0.694250 | 0.725989 |
| 5 | 0.557797 | 0.754054 | 1.313641 | 0.638418 |
| 6 | 0.928897 | 0.721622 | 1.892271 | 0.638418 |
| 7 | 0.777449 | 0.732432 | 0.691914 | 0.700565 |
| 8 | 0.669376 | 0.771622 | 1.070720 | 0.632768 |
| 9 | 0.557235 | 0.789189 | 1.074323 | 0.646893 |
| 10 | 0.881386 | 0.736486 | 0.768388 | 0.728814 |
| 11 | 0.586019 | 0.793243 | 0.733242 | 0.748588 |




We chose to plot the graph for VGG16 model because the average val_accuracy is 83+% which is the best among all the models

## Prediction using VGG16 model

# CONCLUSION

- Key Findings and Conclusions of the Study

  The VGG 16 model is able to classify the images into correct categories with a high average accuracy of 83+%

- Learning Outcomes of the Study in respect of Data Science

  1. This model's accuracy can be increased significantly with scraping more images and enlarging the dataset.
  2. SoftMax as the activation function and the Adam optimizer worked amazingly well with categorizing the dataset and

- Limitations of this work and Scope for Future Work

  Limitation: Larger dataset can significantly improve the accuracy.

  Future Scope: There has been a lot of research going on in the deep learning sector and new research papers have been pointedly enhancing the neural networks and their applications in the real world. Early work on trend analysis [30] broke down catwalk images from NYC fashion shows to find style trends in high-end fashion. Recent advance in deep learning enabled more work on this area. [31–33] utilized deep networks to extract clothing attributes from images and created a visual embedding of clothing style cluster in order to study the fashion trends in clothing around the globe.