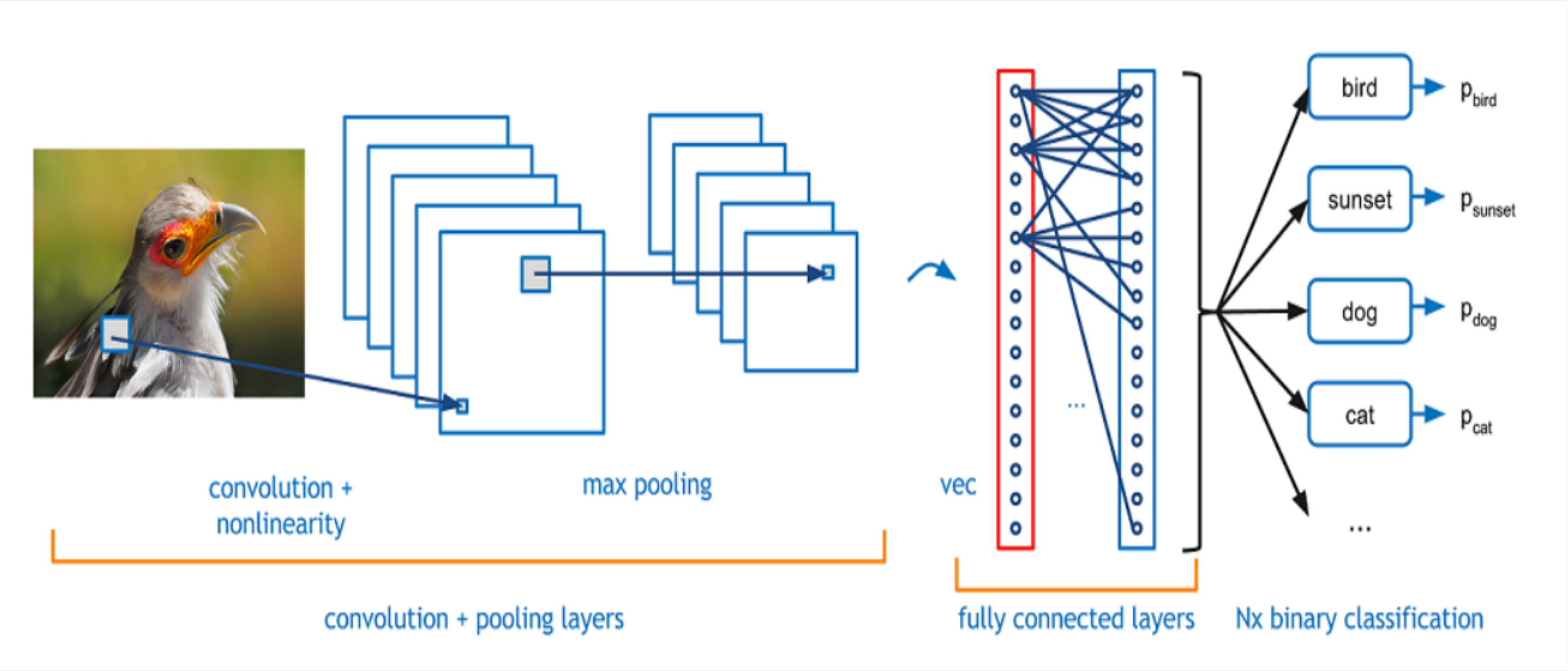
Image Classification Using CNN On CIFAR10

CONTENTS

Objective

Ninad Gadre

Creating a Convolution Neural Network for Image Classification



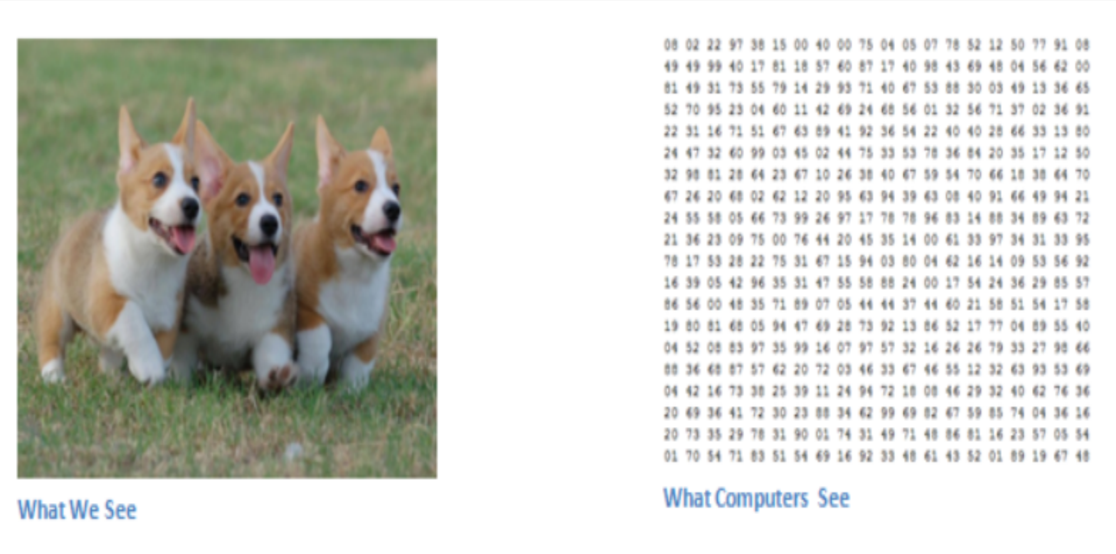
Introduction

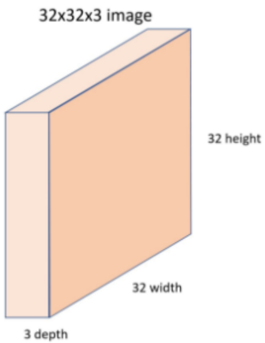
Problem Space

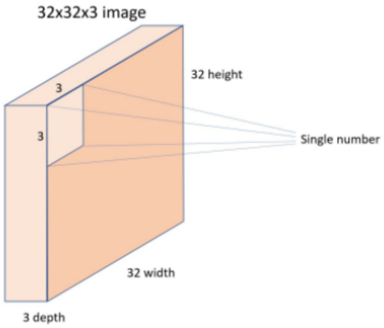
Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image.

When we see an image, most of the time we are able to immediately characterize the scene and give each object a label

These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different environments are ones that we do share with our fellow machines







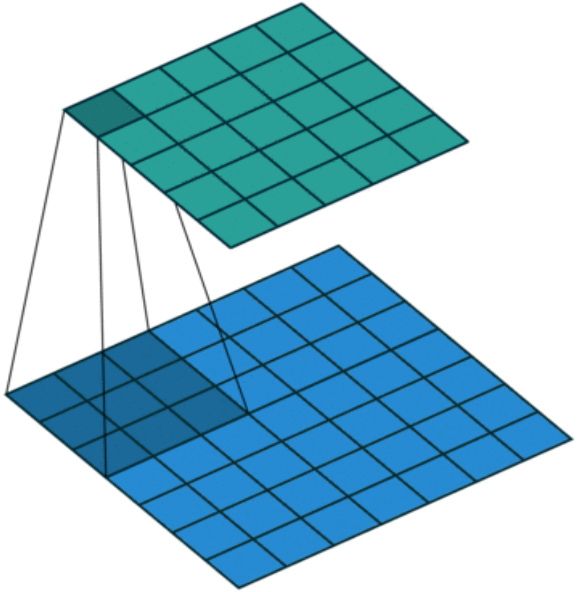


Image (x)

4	3	8
2	8	7
0	8	4

Filter (A) B = 0

9	-4	6
8	1	-1
-6	7	-3

Result

$$(4 * 9) + (3 * -4) + (8 * 6) +$$
$$(2 * 8) + (8 * 1) + (7 * -1) +$$
$$(0 * -6) + (8 * 7) + (4 * -3) + 0$$

$$= \sum x_n * A_n + B$$
$$= 133$$

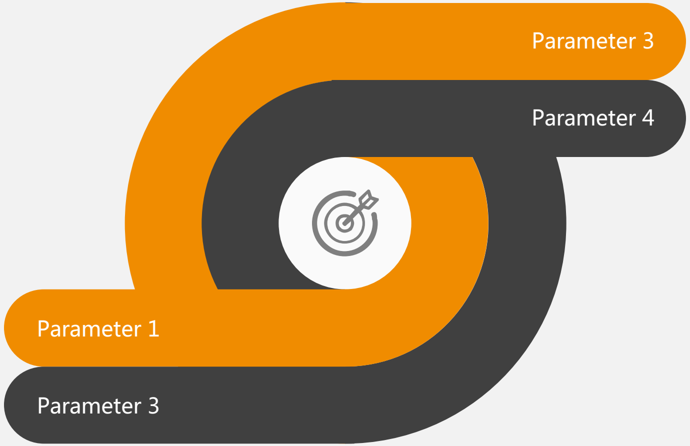
Choosing the Hyperparameter

01

How many conv layers?

02

Which activation layer?



03

What will be the filter size?

04

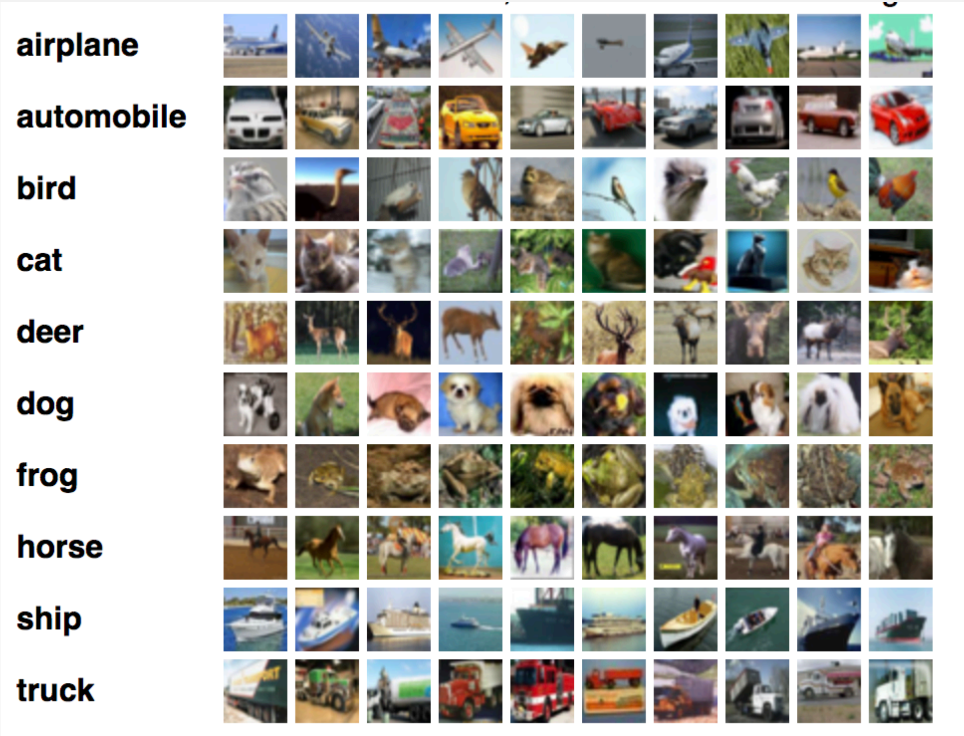
Values for strides and padding?

Data Set & Tool

Xinyun Chen

CIFAR 10

- 60000 32x32 color images in 10 classes
- 5 training batches + 1 test batch each with 10000 images
- test batch: 1000 randomly-selected images from each class training batches: 5000 images from each class



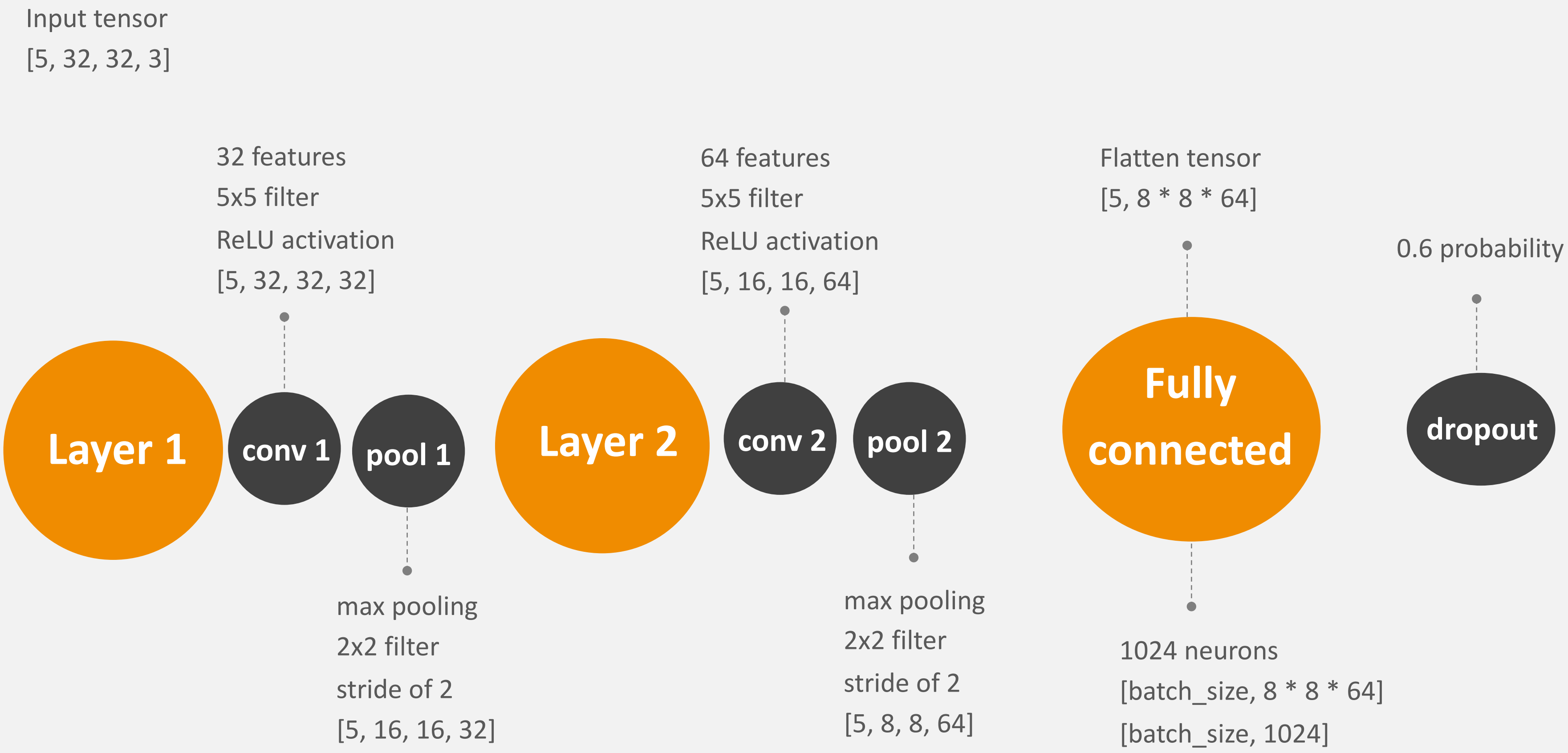
Data Processing

1. Reshape as (10000,3,32,32)
   Transpose as (0,2,3,1)
2. Normalize pixel values
3. Floating point values
4. Store them in array
5. Concatenate into table in column order

APPROACH

Implemented Convolutional Neural Network on the dataset using Keras & TensorFlow

CNN Model



Input tensor
[5, 32, 32, 3]
32 features
5x5 filter
ReLU activation
[5, 32, 32, 32]
64 features
5x5 filter
ReLU activation
[5, 16, 16, 64]
Flatten tensor
[5, 8 * 8 * 64]
0.6 probability
Layer 1
conv 1
pool 1
Layer 2
conv 2
pool 2
Fully connected
dropout
max pooling
2x2 filter
stride of 2
[5, 16, 16, 32]
max pooling
2x2 filter
stride of 2
[5, 8, 8, 64]
1024 neurons
[batch_size, 8 * 8 * 64]
[batch_size, 1024]

Results

Batch Size : 128

Steps: 2000

5 * 5 kernel size	3 * 3 kernel size
accuracy: 0.4516	accuracy: 0.6297
loss: 1.5138574	loss: 1.0620928

CNN using Tensorflow

Haimin Zhang

Running Environment: Colab

Develop deep learning applications with Google Colaboratory -on **the free Tesla K80 GPU**- using Keras, Tensorflow and PyTorch

```
!apt-get install -y -qq software-properties-common python-software-properties module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
```

```
!mkdir -p drive
!google-drive-ocamlfuse drive -o nonempty
```

Data Preprocessing

01

Randomly crop a [24, 24] section of the image

02

Randomly flip the image horizontally

03

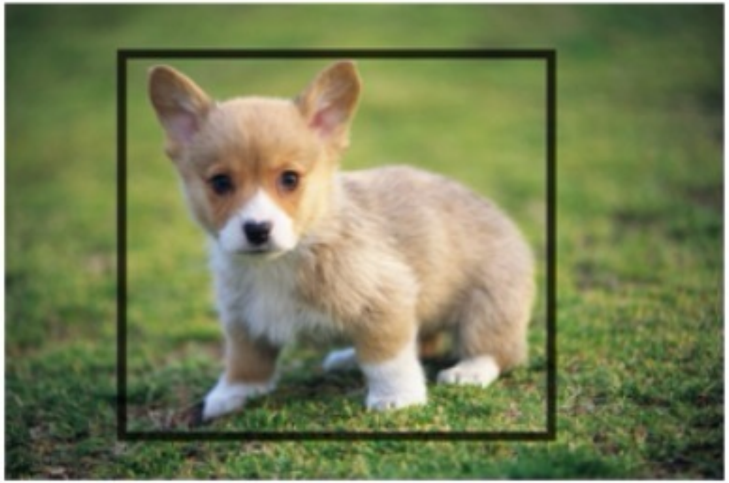
Adjust the brightness of images by a random factor

04

Adjust the contrast of an image by a random factor

01

Randomly crop a [24, 24] section of the image



02

Randomly flip the image horizontally

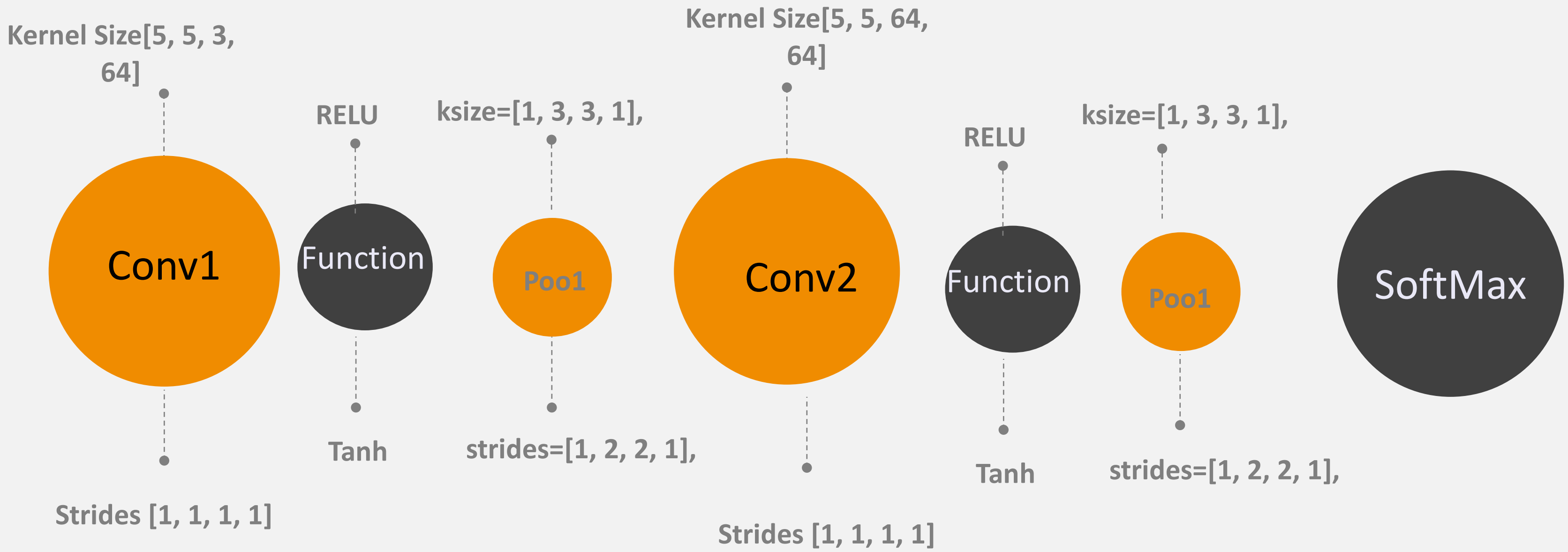
03

Adjust the brightness of images by a random factor

04

Adjust the contrast of an image by a random factor

Modeling



Kernel Size[5, 5, 3, 64]
Conv1
Strides [1, 1, 1, 1]
RELU
Function
Tanh
ksize=[1, 3, 3, 1],
Poo1
strides=[1, 2, 2, 1],
Kernel Size[5, 5, 64, 64]
Conv2
Strides [1, 1, 1, 1]
RELU
Function
Tanh
ksize=[1, 3, 3, 1],
Poo1
strides=[1, 2, 2, 1],
SoftMax

Results

RELU activation function

Precision :0.836

Loss: 0.89

Tanh activation function

Precision :0.785

Loss: 0.81

CNN using Keras

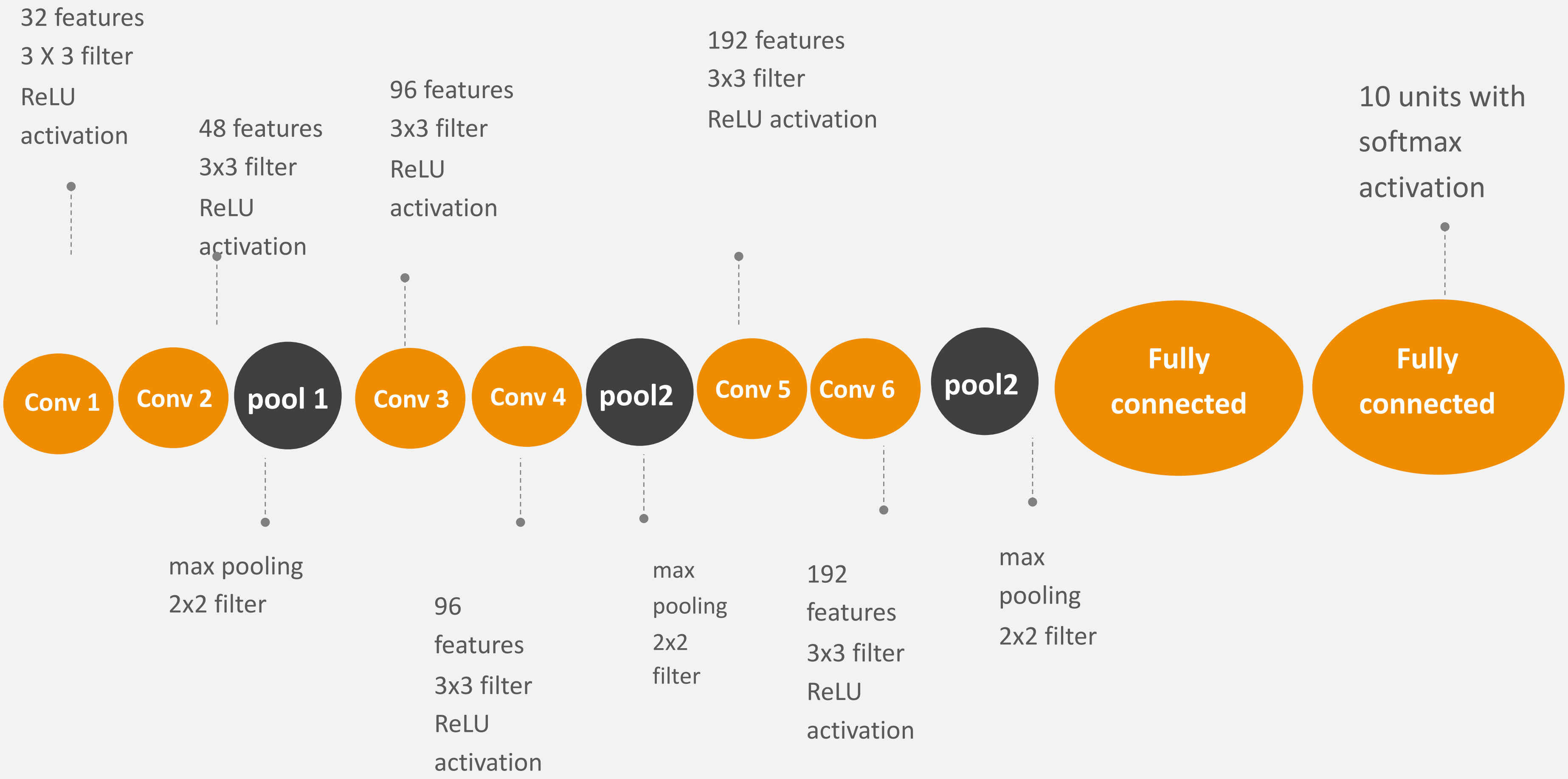
Krutika Deshpande

Keras is an open source neural network Python library which can run on top of other machine learning libraries like TensorFlow, CNTK or Theano. It allows for an easy and fast prototyping, supports convolutional, recurrent neural networks and a combination of the two

Parameters for the CNN built:

Batch Size = 128
Number of Classes = 10
epochs = 100
optimizer = "adam"
learning_rate = 0.01
loss = 'categorical_crossentropy

CNN Model


32 features
3 X 3 filter
ReLU
activation
Conv 1
Conv 2
48 features
3x3 filter
ReLU
activation
pool 1
max pooling
2x2 filter
96 features
3x3 filter
ReLU
activation
Conv 3
Conv 4
96
features
3x3 filter
ReLU
activation
pool2
max
pooling
2x2
filter
192 features
3x3 filter
ReLU activation
Conv 5
Conv 6
192
features
3x3 filter
ReLU
activation
pool2
max
pooling
2x2 filter
Fully
connected
Fully
connected
10 units with
softmax
activation

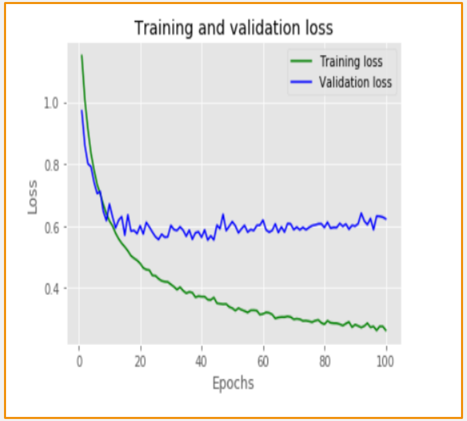
Results

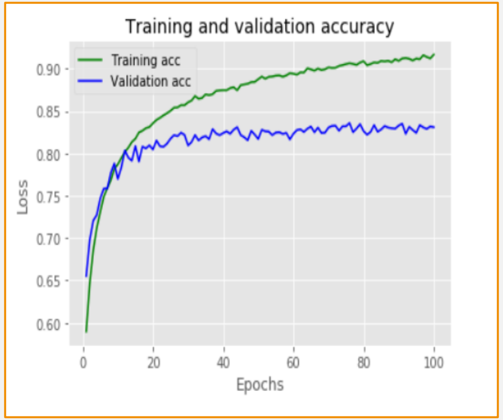
Batch Size : 128

Epochs : 100

Relu activation	Tanh activation
accuracy: 0.8308 loss: 0.6233	accuracy: 0.8289 loss: 0.6324

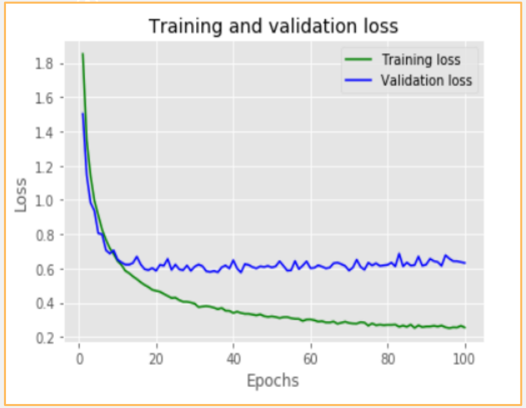
Activation Function : Relu



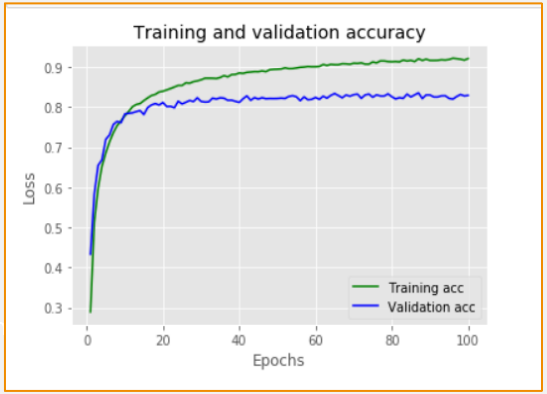


TVL:0.623L:0.2642

TA: 91.66 VA: 83.08



TL:0.2557 VL:0.6324



TA: 92.07 VA: 82.89

Activation Function : TanH

Conclusion

Ninad Gadre

Extensions To Improve Model Performance

Deeper Network Topology. The larger network presented is deep, but larger networks could be designed for the problem. This may involve more feature maps closer to the input and perhaps less aggressive pooling. Additionally, standard convolutional network topologies that have been shown useful may be adopted and evaluated on the problem.

Train for More Epochs. Each model was trained for a very small number of epochs, 25. It is common to train large convolutional neural networks for hundreds or thousands of epochs. I would expect that performance gains can be achieved by significantly raising the number of training epochs

Apply sparse-encoding
Sparse representations are effective for storing patterns and maximizing the independence of features this would lead to more pronounced identification of complex image features

Image Data Augmentation. The objects in the image vary in their position. Another boost in model performance can likely be achieved by using some data augmentation. Methods such as standardization and random shifts and horizontal image flips may be beneficial.

Transfer Learning
Huge CNNs and large input images can take weeks on end to train. Luckily many world famous CNNs such as Google's Inception V3 and Microsoft's Resnet from the ImageNet competition, can be used to generate your own models using some relatively computationally cheap methods.