Batch Normalization Cifar-10 Dataset

FADILLA
ZENNIFA
ファディラゼニファフ

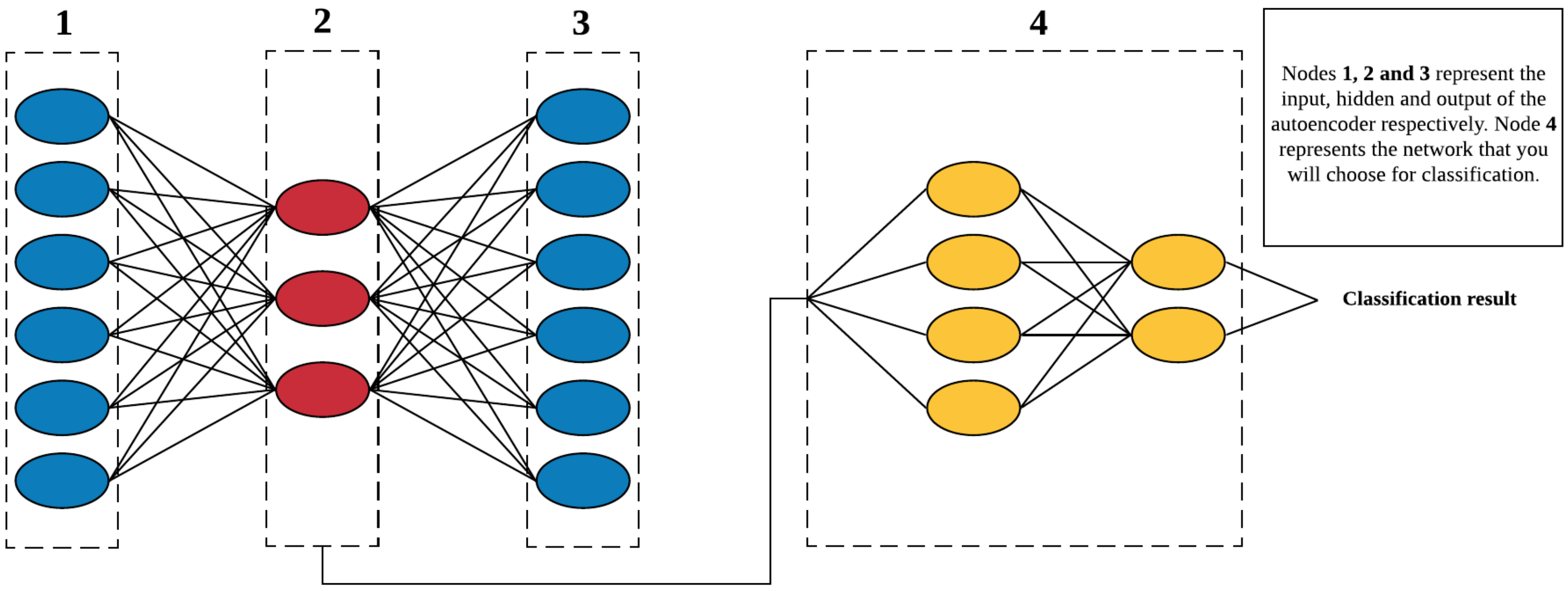
hiring process
Ridge-i

Aim

1. Design a network that combines supervised and unsupervised architectures in one model to achieve a classification task.
2. Find the optimum architecture
3. Find appropriate batch

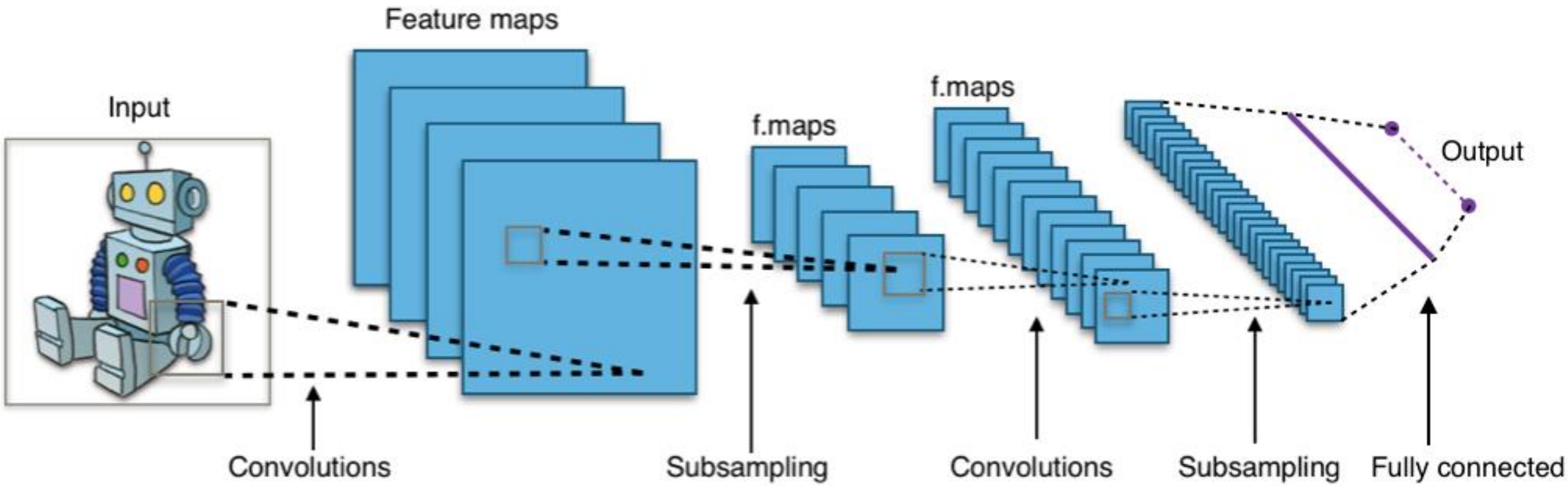
Introduction

? Establish the best design



llustration from Ridge-I company hiring process

CNN (Convolutional neural network)



https://commons.wikimedia.org/wiki/File:Typical_cnn.png

Batch size, Epoch, Layers

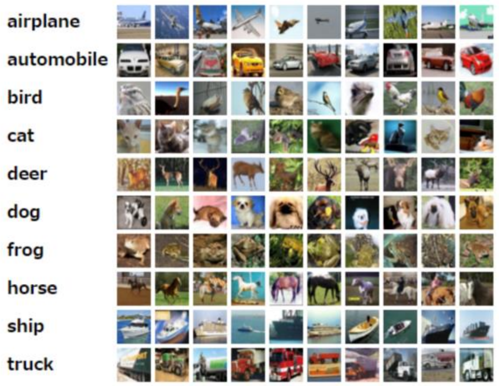
Batch size is refers to the number of training examples utilized in one iteration.

An **epoch** indicates the number of passes through the entire training dataset the **machine learning** algorithm has completed

Layer a collection of 'nodes' operating together at a specific depth within a **neural network (NN)**.

Batch Normalization implemented during training by calculating the mean and standard deviation of each input variable to a layer per mini-batch and using these statistics to perform the standardization.

Methodology



Cifar10 Dataset

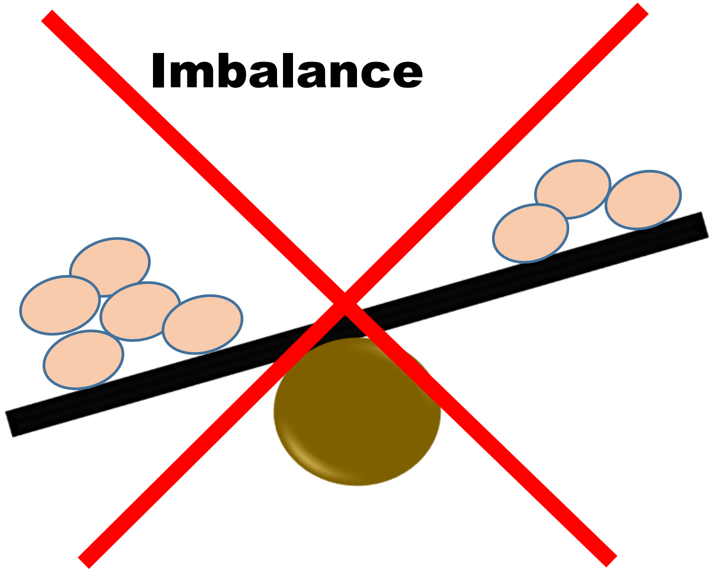
PC Specification

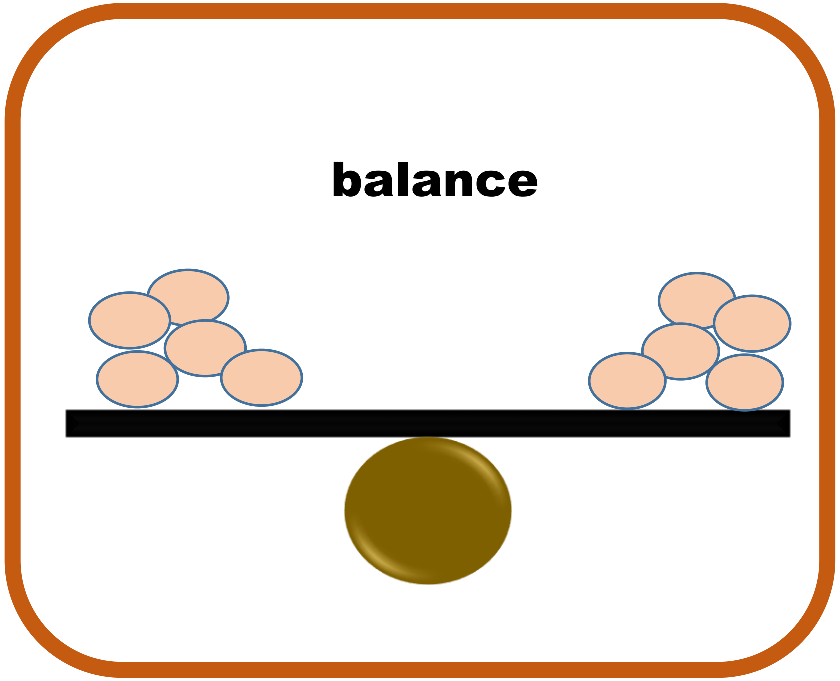
1. Processor Intel® Core™ i5
   Operating System Windows 10
2. Graphics NVIDIA® GeForce® 940 MX(Optional)
3. Memory 8 GB DDR4
4. CPU 64 bit

Methodology

Classes in Cifar 10

0 = airplan,
1= automobile,
2 =bird,
3= cat,
4= deer,
5= dog,
6 = frog,
7= horse,
8= ship,
9 = truck.



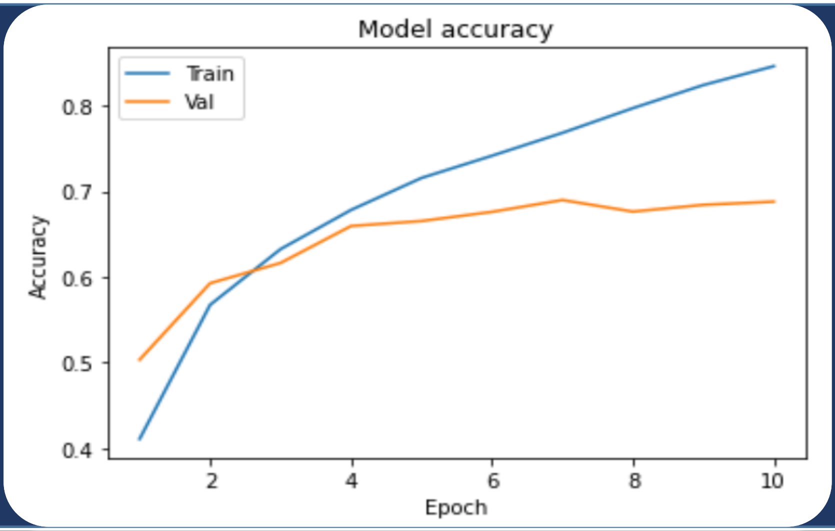


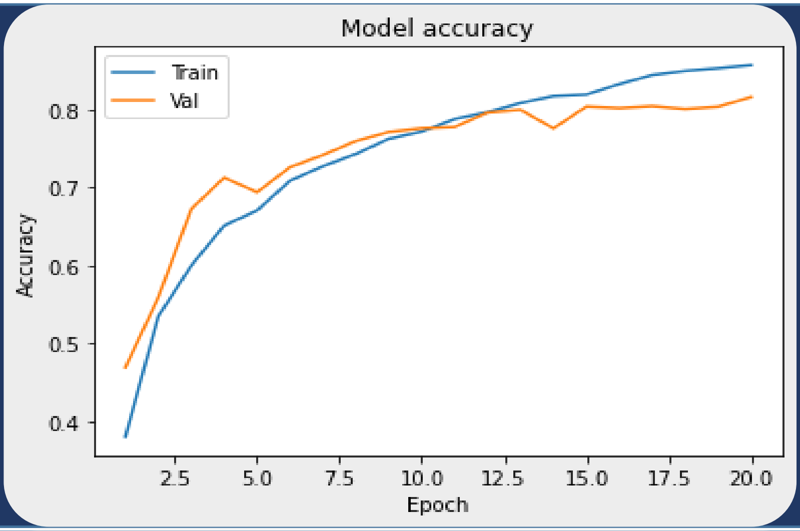
Criteria

Training set of Cifar 10 consists:
50% of Bird [2]
50% of Deer [4]
50% of Truck [9]

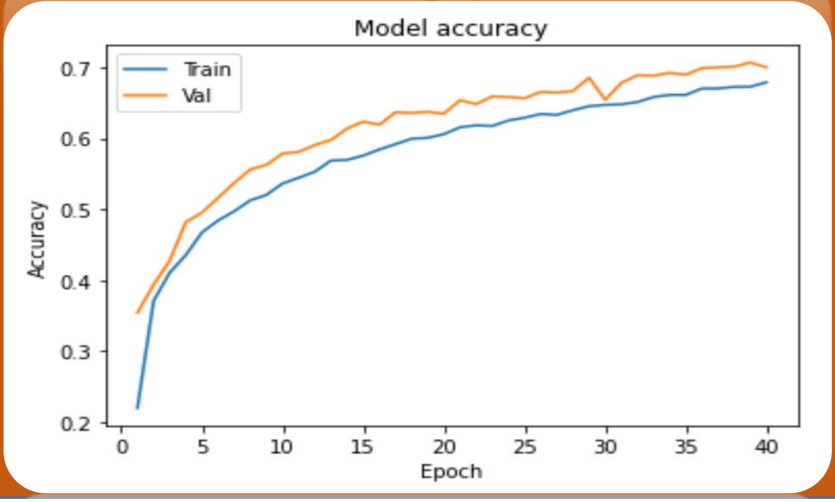
I also chose other class 50%

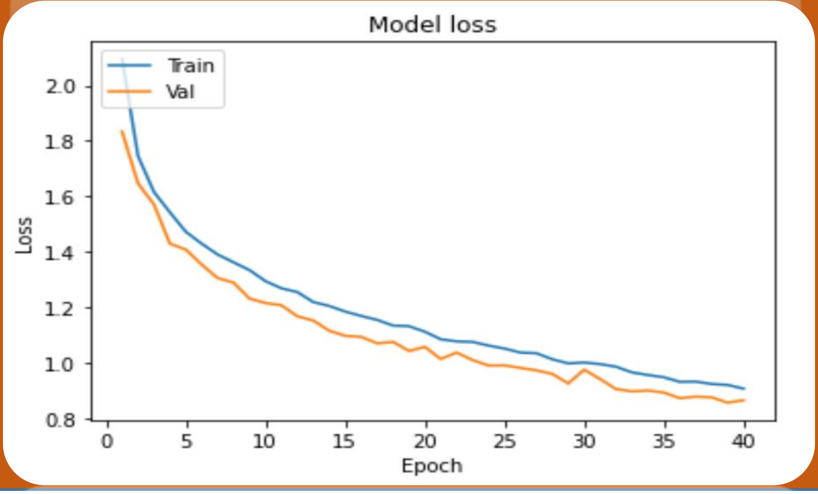
Result





Overfitting





Underfitting

Layers

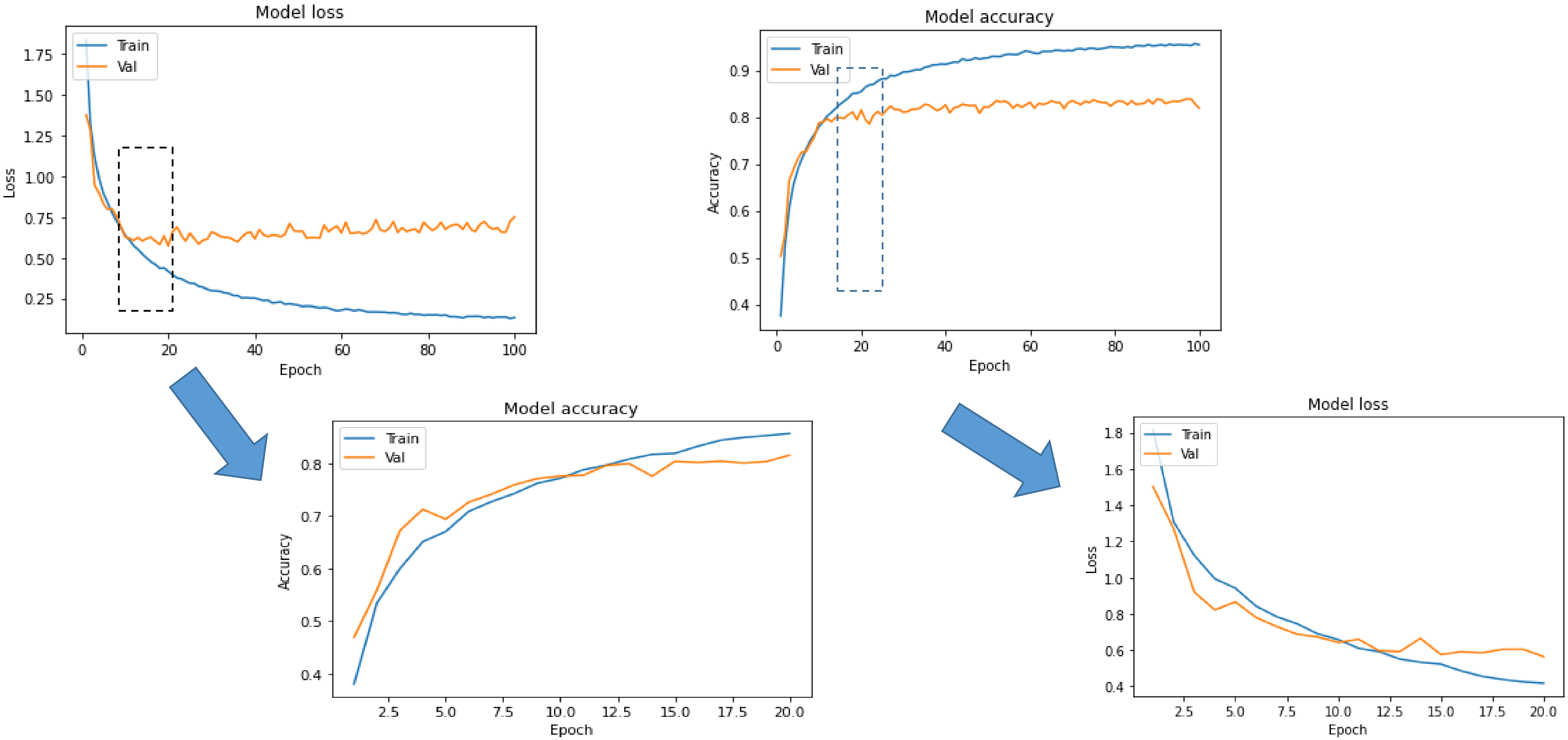
Number of layer	Layer performance	
	Training	Testing
3	96.04%	61.88%
3+ max pooling	91.82%	69.24%
5+ max pooling	77.80%	75.49%
6+ max pooling	95.42%	81.96%

Batch

Number of Epoch	Batch sizes	Batch performance	
		Training	Testing
25	10	68.44%	67.66%
10	32	84.59%	68.78%
100	64	87.00%	75.87%
40	502	67.90%	70.02%
100	512	77.80%	75.49%
10	1024	48.19%	52.69%
10	2048	43.69%	45.15%
20	Normalization	85.71%	81.59%
100	Normalization	95.42%	81.96%

The Best performance

Selected model



Application in New data

In [26]:
```
img = tf.read_file("Aeroplan.jpg")
img = tf.image.decode_jpeg(img, channels=3)
img.set_shape([None, None, 3])
img = tf.image.resize_images(img, (32, 32))
img = img.eval(session=sess) # convert to numpy array
img = np.expand_dims(img, 0) # make 'batch' of 1
# prepare pixel data
img = img.astype('float32')
img = img / 255.0

#pred = model.predict(img)
result = new_model.predict_classes(img)
print(result[0])
Image("Aeroplan.jpg")
```

```
0
```

Out[26]:

In [14]:
```
#evaluate model
from IPython.display import Image
Image("deer.png")
#print('Images Shape: {}'.format(X_train.shape))
```

Out[14]:

In [15]:
```
init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)


img = tf.read_file("deer.png")
img = tf.image.decode_jpeg(img, channels=3)
img.set_shape([None, None, 3])
img = tf.image.resize_images(img, (32, 32))
img = img.eval(session=sess) # convert to numpy array
img = np.expand_dims(img, 0) # make 'batch' of 1
# prepare pixel data
img = img.astype('float32')
img = img / 255.0

#pred = model.predict(img)
result = new_model.predict_classes(img)
print(result[0])
#acc

#class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
  #              'dog', 'frog', 'horse', 'ship', 'truck']
```

```
WARNING:tensorflow:From /home/agemono/.pyenv/versions/anaconda3-4.4.0/lib/python3.6/site-packages/tensorflow_core/python/util/tf_should_use.py:198: initialize_all_variables (from tensorflow.python.ops.variables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
4
```

Conclusion

Normalize batch > Manual batch

6 layers > less than 6 layers

References

- The Y. Weng, T. Zhou, L. Liu and C. Xia, "Automatic Convolutional Neural Architecture Search for Image Classification Under Different Scenes," in *IEEE Access*, vol. 7, pp. 38495-38506, 2019.
- C. Yu, X. He, H. Ma, X. Qi, J. Lu and Y. Zhao, "S-DenseNet: A DenseNet Compression Model Based on Convolution Grouping Strategy Using Skyline Method," in IEEE Access, vol. 7, pp. 183604-183613, 2019.
- P. Panda, I. Chakraborty and K. Roy, "Discretization Based Solutions for Secure Machine Learning Against Adversarial Attacks," in IEEE Access, vol. 7, pp. 70157-70168, 2019.
- A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 Dataset, [online] Available: https://www.cs.toronto.edu/kriz/cifar.htm