Classification of MRI images using CNN for tumour detection

(Part 2 - Analysis of the training and validation)

Comparision of training and validation loss and accuracy

Let us load the training history of the different models using Pandas

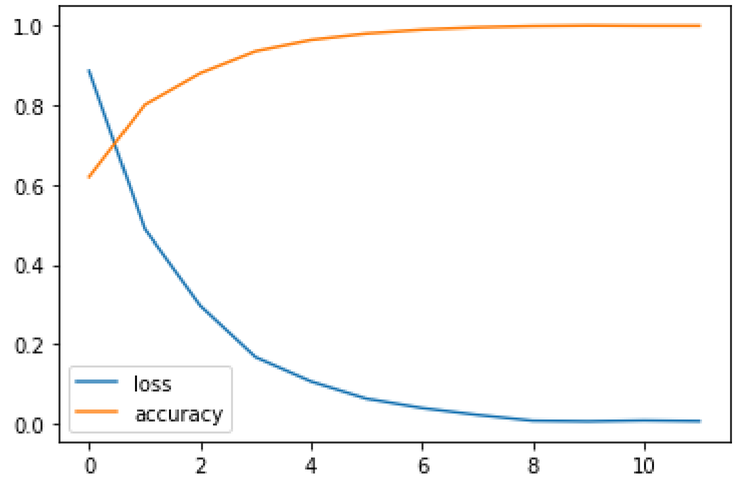
```
import pandas as pd

basic_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/basic.csv")
res_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/res.csv")
vgg_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/vgg.csv")
mobile_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/mobile.csv")
inception_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/inception.csv")
alexnet_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/alexnet.csv")
lenet_df = pd.read_csv("/content/drive/MyDrive/mri-cnn/lenet.csv")
```

Model 1 - Basic CNN

Training Loss vs Accuracy

```
basic_df[["loss","accuracy"]].plot()
```

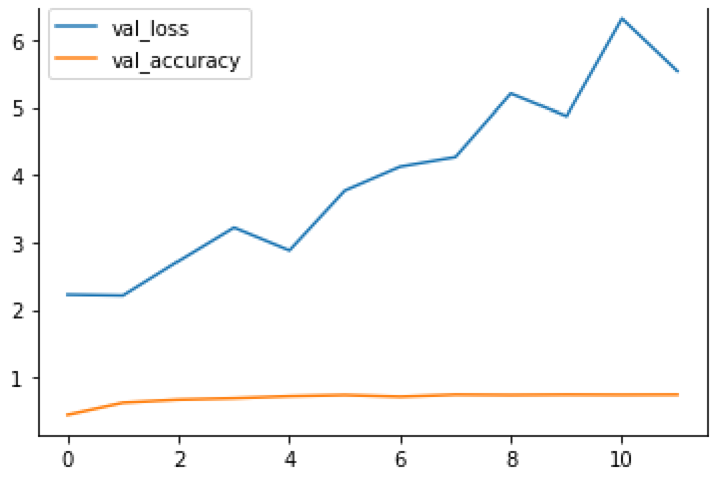
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1daf3ec50>
```



Validation loss vs accuracy

```
basic_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1daf3e550>
```

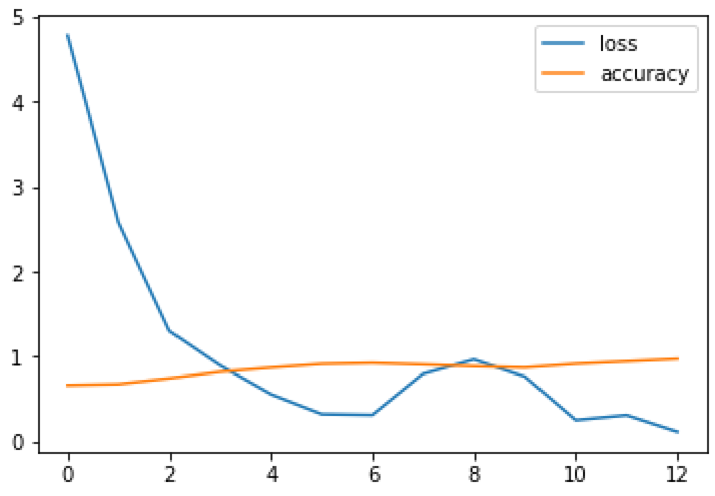


Model 2 - ResNet50

Training Loss vs Accuracy

```
res_df[["loss","accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da90e910>
```



Validation loss vs accuracy

```
res_df[["val_loss","val_accuracy"]].plot()
```

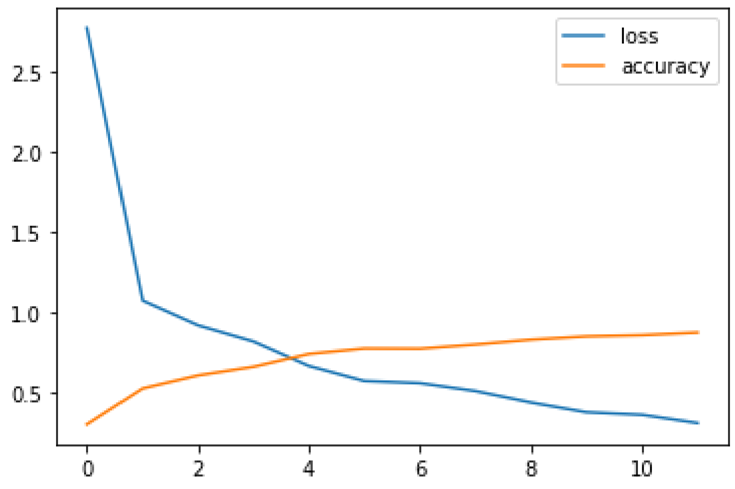
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da88d390>
```

Model 3 - VGG16

Training Loss vs Accuracy

```
vgg_df[["loss","accuracy"]].plot()
```

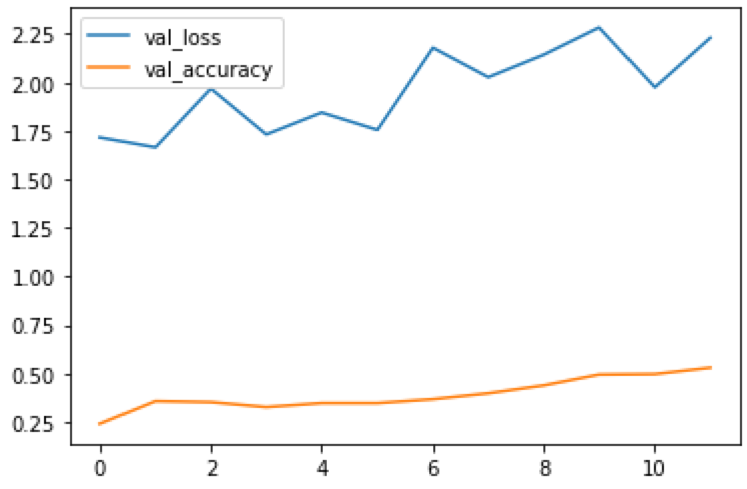
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da811190>
```



Validation loss vs accuracy

```
vgg_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da7915d0>
```

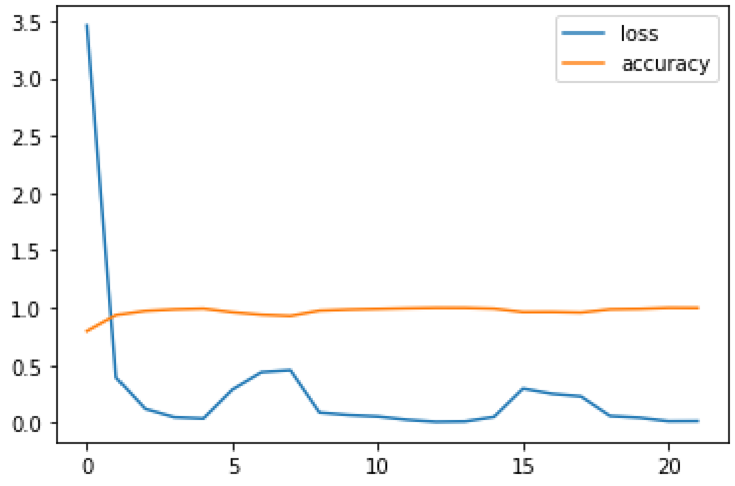


Model 4 - MobileNet

Training Loss vs Accuracy

```
mobile_df[["loss","accuracy"]].plot()
```

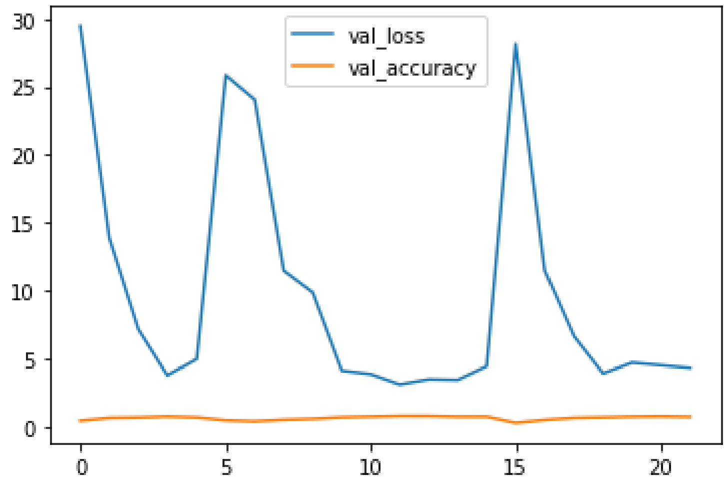
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1e1117c50>
```



Validation loss vs accuracy

```
mobile_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da760750>
```



Model 5 - InceptionNet v2

Training Loss vs Accuracy

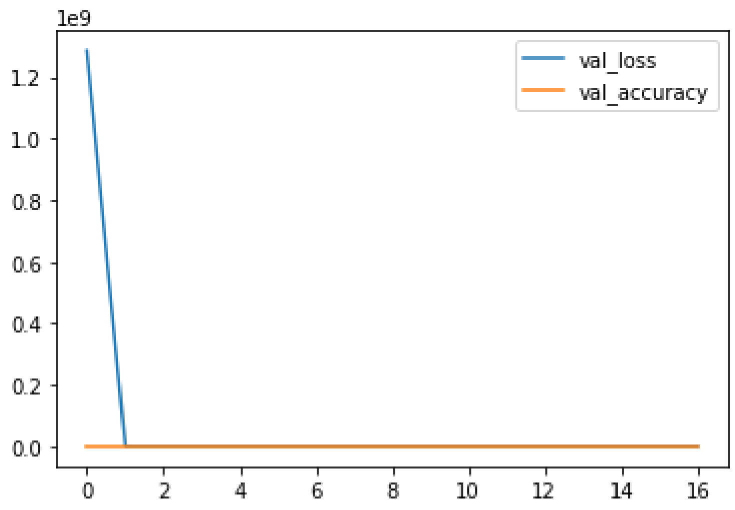
```
inception_df[["loss","accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da783b90>
```

Validation loss vs accuracy

```
inception_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da5f53d0>
```

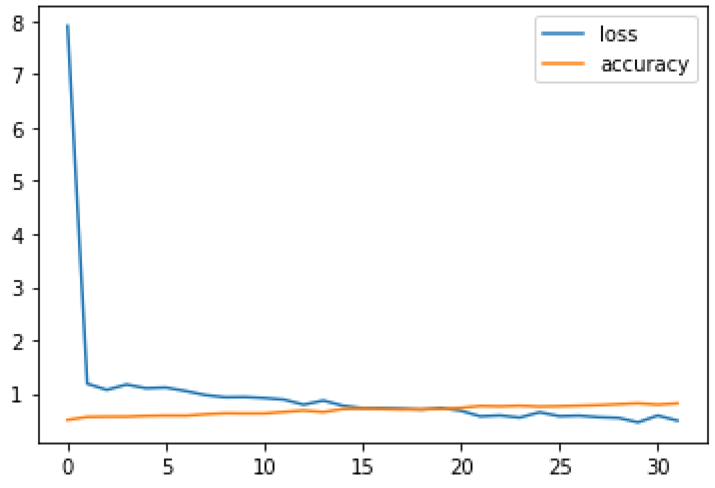


Model 6 - AlexNet

Training Loss vs Accuracy

```
alexnet_df[["loss","accuracy"]].plot()
```

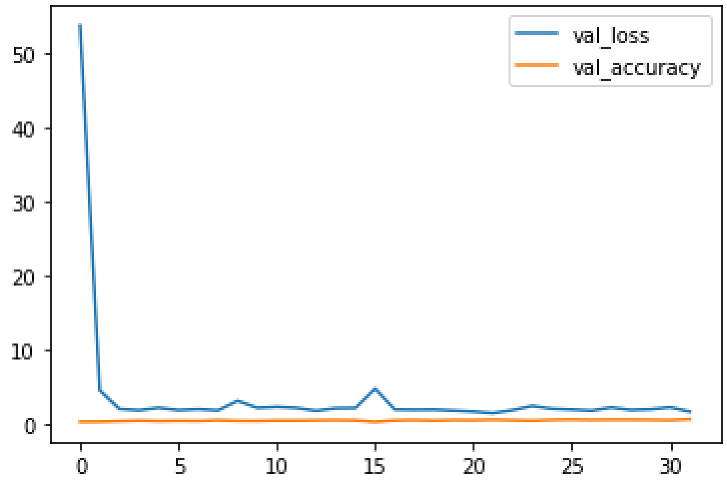
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da4fb850>
```



Validation loss vs accuracy

```
alexnet_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da48c350>
```

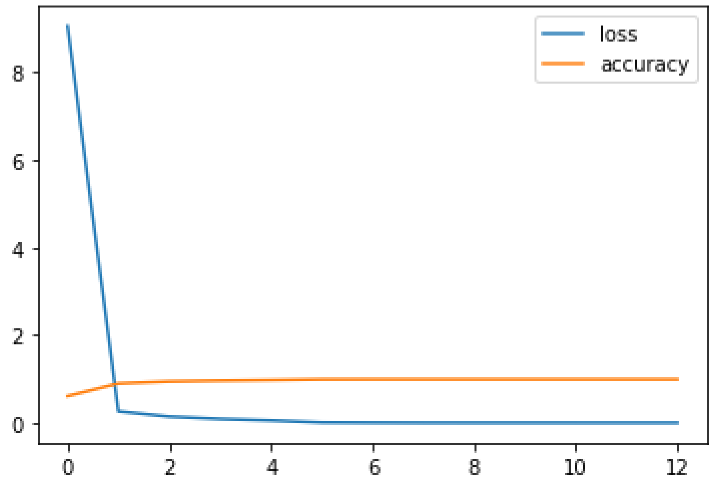


Model 7 - LeNet

Training Loss vs Accuracy

```
lenet_df[["loss","accuracy"]].plot()
```

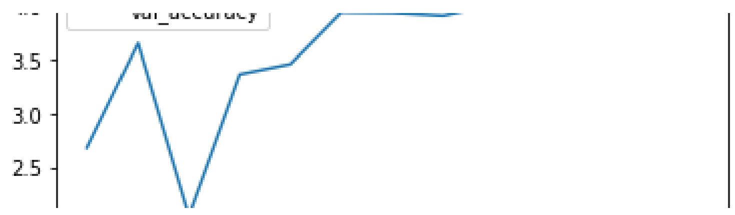
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da40a810>
```



Validation loss vs accuracy

```
lenet_df[["val_loss","val_accuracy"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da3d2290>
```



Comparisions across different models

Merge the training histories of the different models into a single DataFrame object.

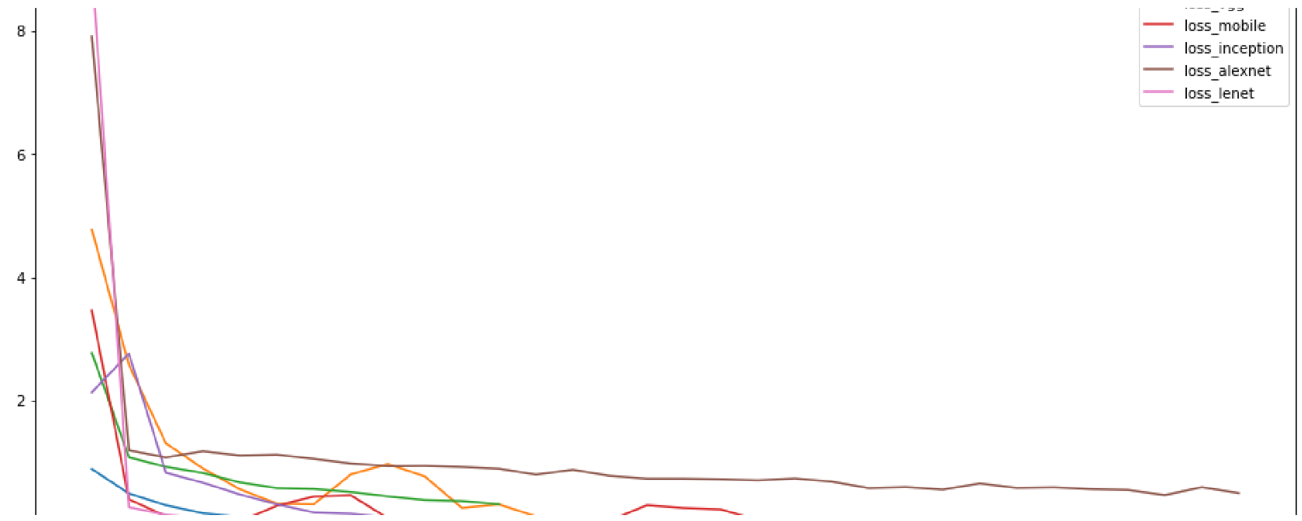
```
result = basic_df.merge(res_df, on=['Unnamed: 0'], suffixes=('_basic', '_resnet'), how='ou
result = result.merge(vgg_df, on=['Unnamed: 0'], how='outer')
result = result.merge(mobile_df, on=['Unnamed: 0'], suffixes=('_vgg', '_mobile'), how='out
result = result.merge(inception_df, on=['Unnamed: 0'], how='outer')
result = result.merge(alexnet_df, on=['Unnamed: 0'], suffixes=('_inception', '_alexnet'),
result = result.merge(lenet_df, on=['Unnamed: 0'], how='outer')
result = result.rename(columns={"loss": "loss_lenet", "accuracy": "accuracy_lenet", "val_l
result = result.drop(["Unnamed: 0"],axis=1)
result.head()
```

	loss_basic	accuracy_basic	val_loss_basic	val_accuracy_basic	loss_resnet	accu
0	0.884231	0.618815	2.224661	0.446701	4.774953	
1	0.488592	0.799652	2.211865	0.626904	2.579997	
2	0.297126	0.878746	2.723235	0.672589	1.302004	
3	0.168282	0.933798	3.221111	0.692893	0.895724	
4	0.107416	0.962021	2.881202	0.723350	0.556298	

Comparision of Training loss

```
result[["loss_basic","loss_resnet","loss_vgg","loss_mobile","loss_inception","loss_alexnet
```

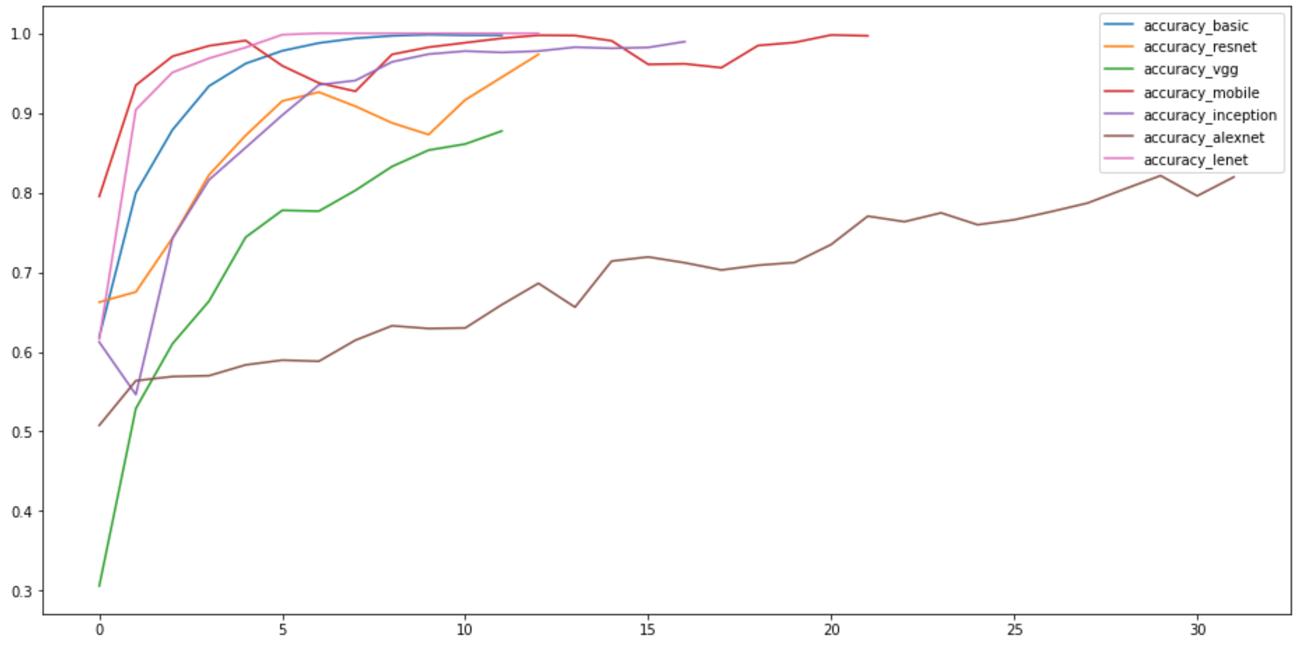
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da3218d0>
```



Comparision of Training accuracy

```
result[["accuracy_basic","accuracy_resnet","accuracy_vgg","accuracy_mobile","accuracy_ince
```

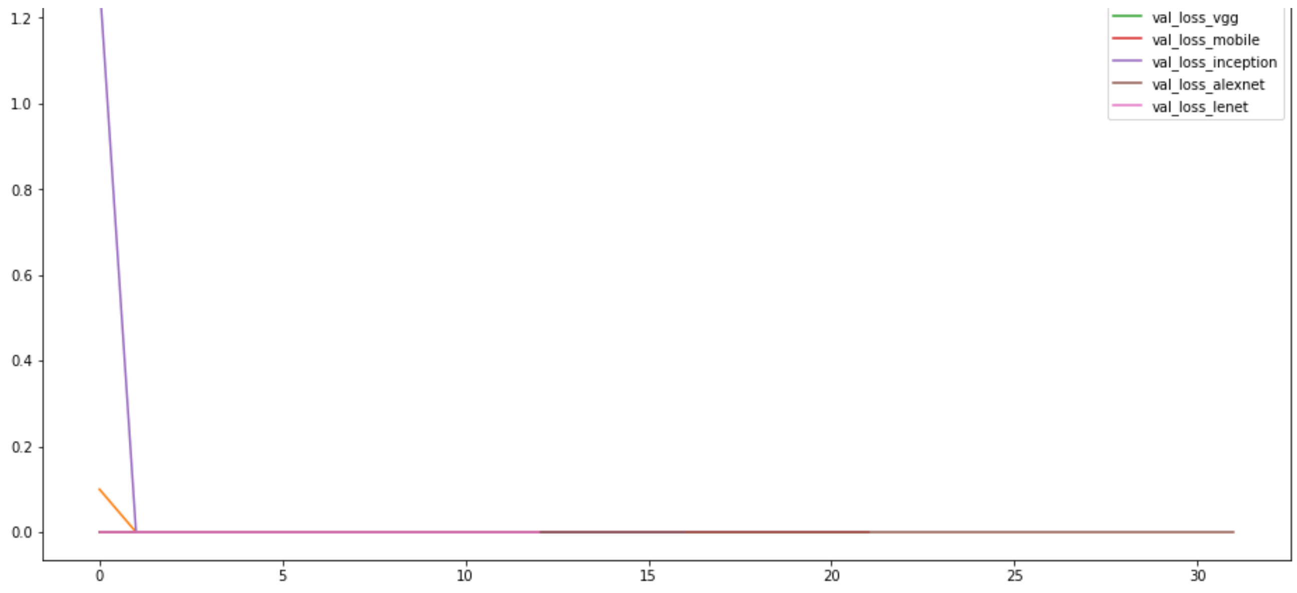
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da1fc7d0>
```



Comparision of Validation loss

```
result[["val_loss_basic","val_loss_resnet","val_loss_vgg","val_loss_mobile","val_loss_ince
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1da1b2b50>
```



Comparision of Validation accuracy

```
result[["val_accuracy_basic","val_accuracy_resnet","val_accuracy_vgg","val_accuracy_mobile
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1d9078890>
```

Conclusion

We see that Basic CNN, LeNet, MobileNet and InceptionNet show high accuracy during the training. Also, ResNet and VGG shows moderate accuracy. However, AlexNet has the worst performance in the training accuracy metric.

However, this observation does not materialize when looking at the validation accuracy. While MobileNet and InceptionNet has shown very high performance in some epochs, the performances are not stable. VGG and AlexNet has shown the worst performance during the validation tests. ResNet has shown moderate performance in some epochs, while the performance has degraded in some other epochs. The best and stable performance are shown by Basic CNN and LeNet during the validation tests.

We can conclude that simpler models like Basic CNN and LeNet are more able to fit the data properly. The larger complcated models (AlexNet, for example) does not perform well in the experiment because the dataset does not have enough variety for the model to train properly, i.e. the model underfitted the data.

Hence, we can try tuning the parameters of a simpler model to achieve higher performance in the dataset.