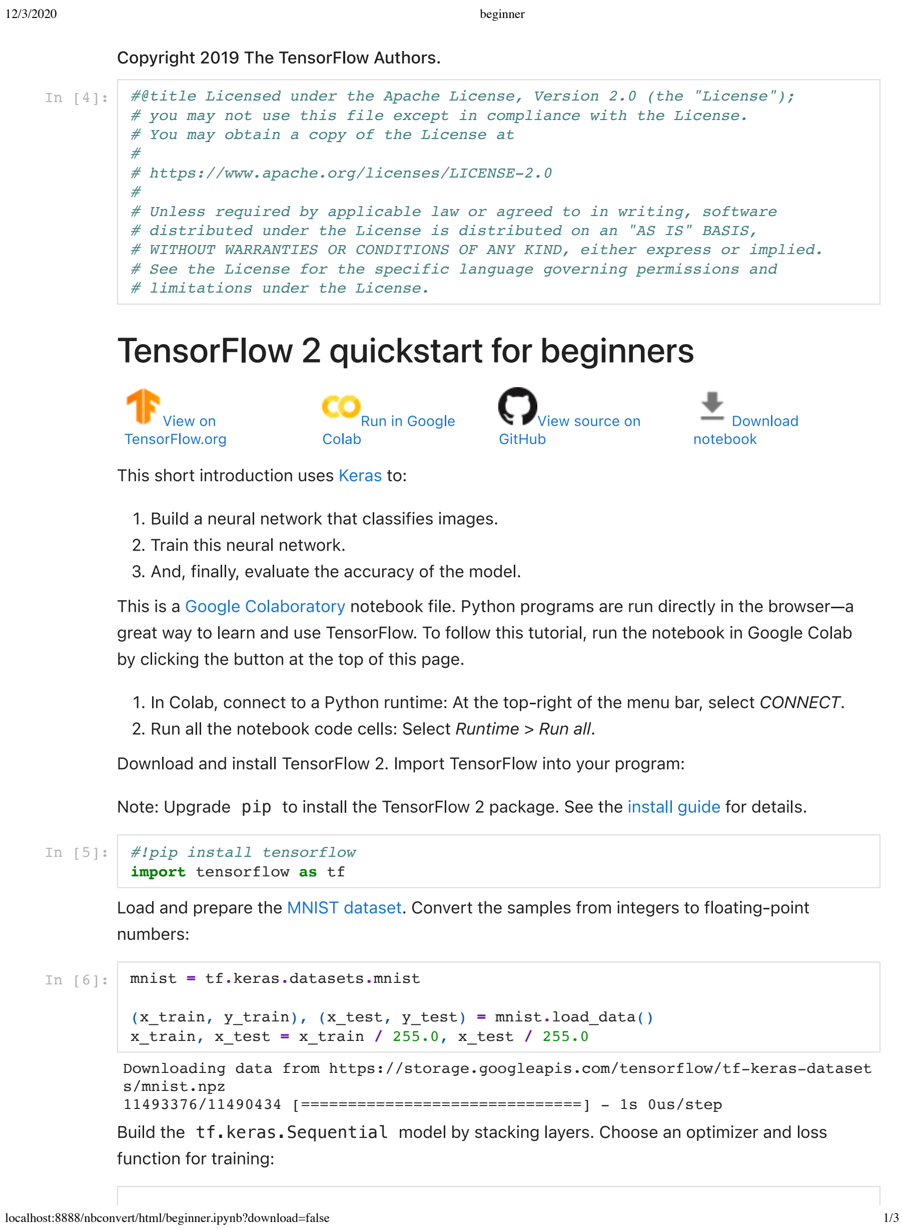
CAP 5636 – Fall 2020 - Homework 3

Step 1. Set up you deep learning environment (2 pts):

To setup a deep Learning environment on mac installed Anaconda. Then installed Tensorflow by using below command:

```
!pip install tensorflow
```

To ensure that we have a working environment, try to run the tensorflow quickstart tutorial: The output of this step is on beginner.pdf file.



12/3/2020 beginner

Copyright 2019 The TensorFlow Authors.

```
In [4]: #@title Licensed under the Apache License, Version 2.0 (the "License");
        # you may not use this file except in compliance with the License.
        # You may obtain a copy of the License at
        #
        # https://www.apache.org/licenses/LICENSE-2.0
        #
        # Unless required by applicable law or agreed to in writing, software
        # distributed under the License is distributed on an "AS IS" BASIS,
        # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
        # See the License for the specific language governing permissions and
        # limitations under the License.
```

TensorFlow 2 quickstart for beginners

View on TensorFlow.org Run in Google Colab View source on GitHub Download notebook

This short introduction uses Keras to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

Download and install TensorFlow 2. Import TensorFlow into your program:

Note: Upgrade `pip` to install the TensorFlow 2 package. See the install guide for details.

```
In [5]: #!pip install tensorflow
        import tensorflow as tf
```

Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

```
In [6]: mnist = tf.keras.datasets.mnist

        (x_train, y_train), (x_test, y_test) = mnist.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
```

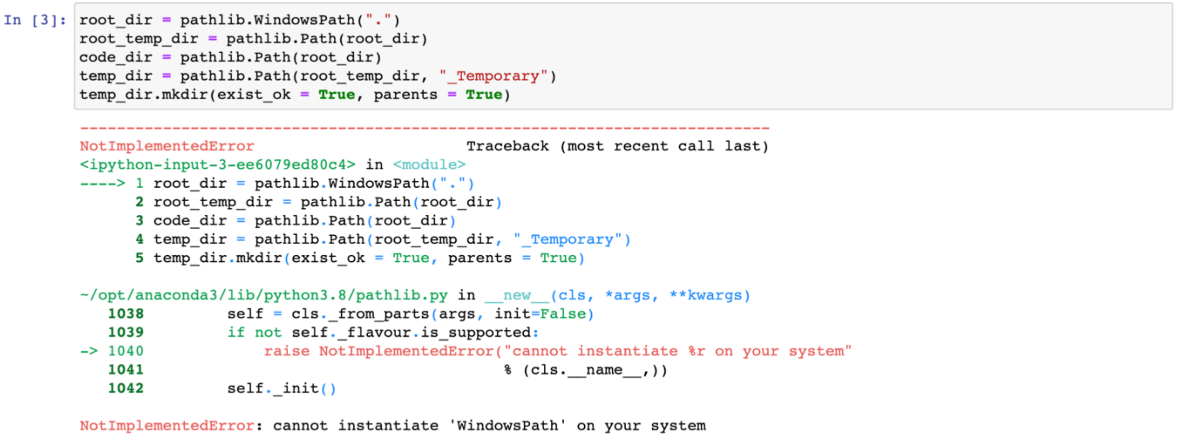
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

Build the `tf.keras.Sequential` model by stacking layers. Choose an optimizer and loss function for training:

localhost:8888/nbconvert/html/beginner.ipynb?download=false 1/3

Step 2: Set up training for catdogmonkey (2 pts):

While performing this step I got below error:



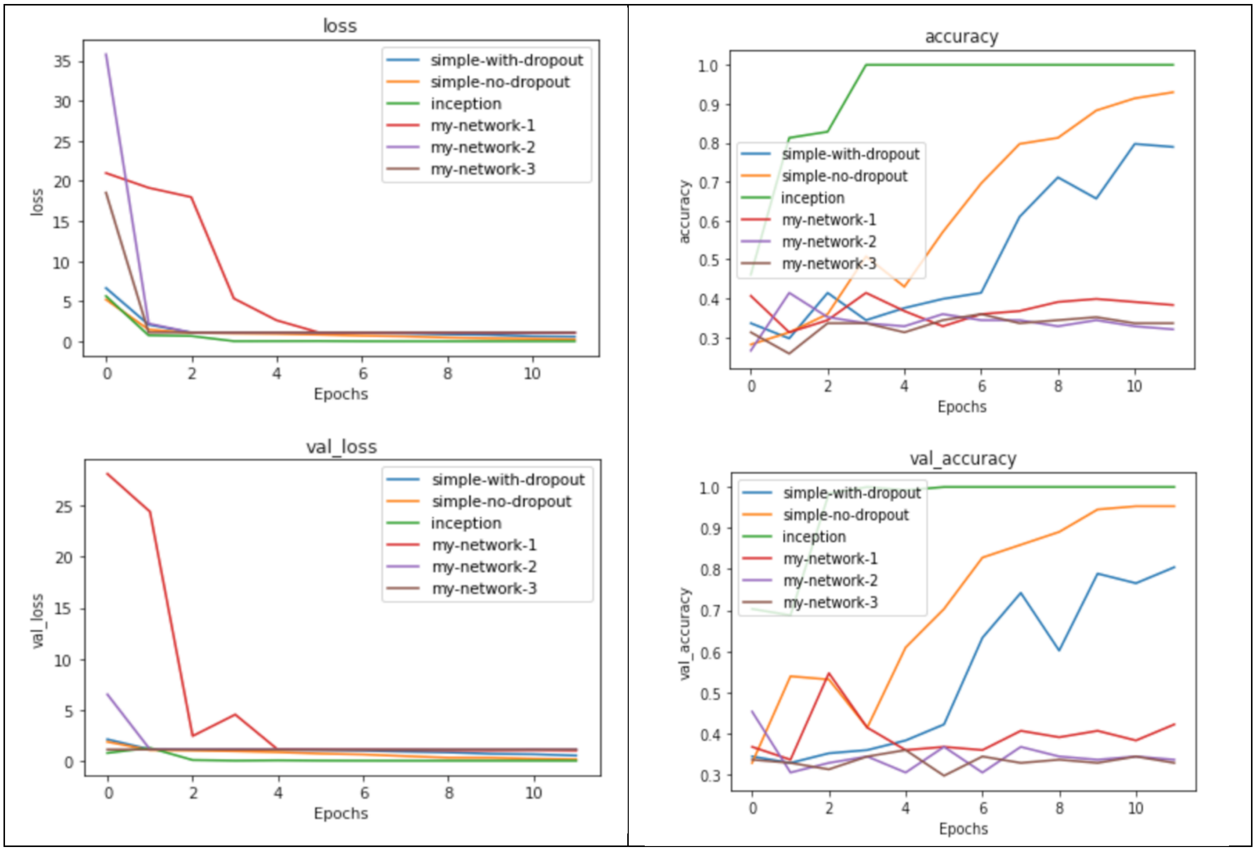
```
In [3]: root_dir = pathlib.WindowsPath(".")
        root_temp_dir = pathlib.Path(root_dir)
        code_dir = pathlib.Path(root_dir)
        temp_dir = pathlib.Path(root_temp_dir, "_Temporary")
        temp_dir.mkdir(exist_ok = True, parents = True)

        ---------------------------------------------------------------------------
        NotImplementedError                       Traceback (most recent call last)
        <ipython-input-3-ee6079ed80c4> in <module>
        ----> 1 root_dir = pathlib.WindowsPath(".")
              2 root_temp_dir = pathlib.Path(root_dir)
              3 code_dir = pathlib.Path(root_dir)
              4 temp_dir = pathlib.Path(root_temp_dir, "_Temporary")
              5 temp_dir.mkdir(exist_ok = True, parents = True)

        ~/opt/anaconda3/lib/python3.8/pathlib.py in __new__(cls, *args, **kwargs)
           1038         self = cls._from_parts(args, init=False)
           1039         if not self._flavour.is_supported:
        -> 1040             raise NotImplementedError("cannot instantiate %r on your system"
           1041                                       % (cls.__name__,))
           1042         self._init()

        NotImplementedError: cannot instantiate 'WindowsPath' on your system
```

To fix this issue, replaced ‘WindowsPath’ with ‘PureWindowsPath’ in CatDogMonkeyHomework.ipynb and ImageDataet.py files.
Below is the output:



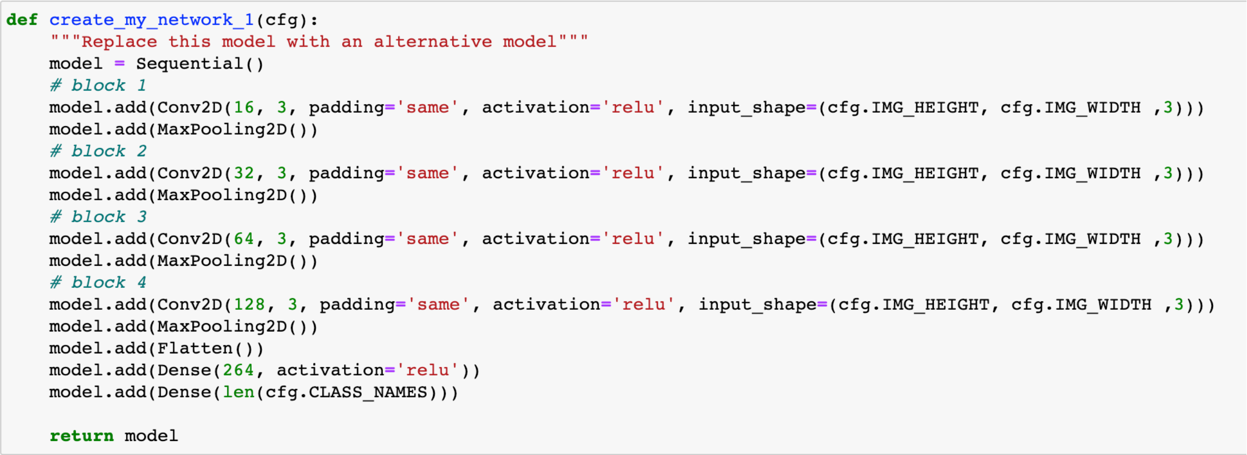
Step 3: Improve the classification (11pts):

There were lots and lots of trials that I have done to get the expected or better outputs. But while documenting those I have documented the turning points of model as well as most successful outcomes.

1. create_my_network_1:

Trial #1: For this function I tried creating 5 layers at very 1st. with Conv2D 16, 32,64,128,1024. But training went very slow and accuracy was around 50%. Which is less.
The training loss kept going up and down, while the validation accuracy, after reaching value = 0.5, did not improve at all. Conclusion: very deep networks with extreme less dense (10) do not train well.

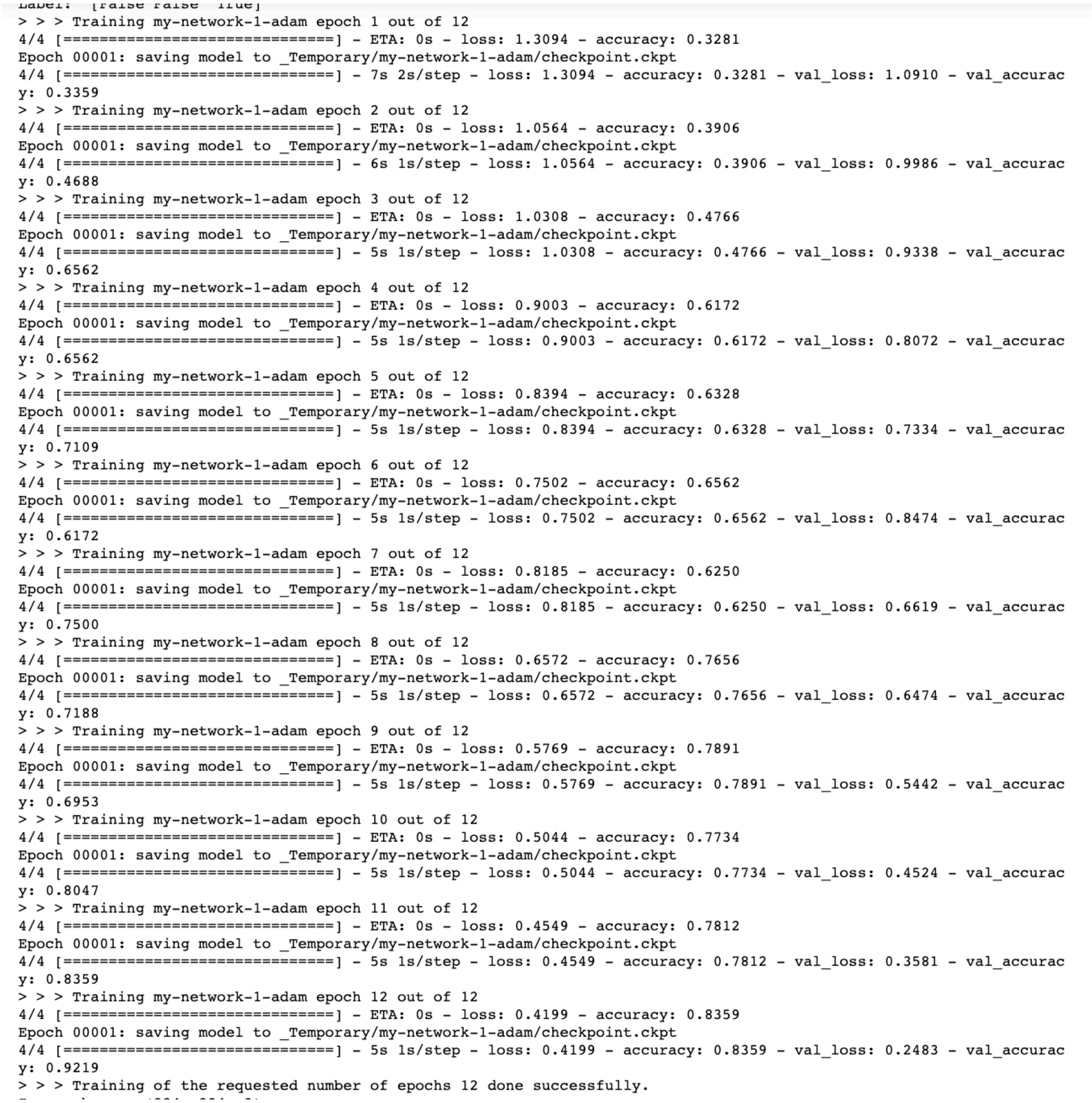
Trial #2: So, then I deleted last layer which was Conv2D 1024 and dense (264) after that training went smooth and got validation accuracy **0.92.**



```
def create_my_network_1(cfg):
    """Replace this model with an alternative model"""
    model = Sequential()
    # block 1
    model.add(Conv2D(16, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 2
    model.add(Conv2D(32, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 3
    model.add(Conv2D(64, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 4
    model.add(Conv2D(128, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    model.add(Flatten())
    model.add(Dense(264, activation='relu'))
    model.add(Dense(len(cfg.CLASS_NAMES)))

    return model
```

The output after 12 epochs is as below:

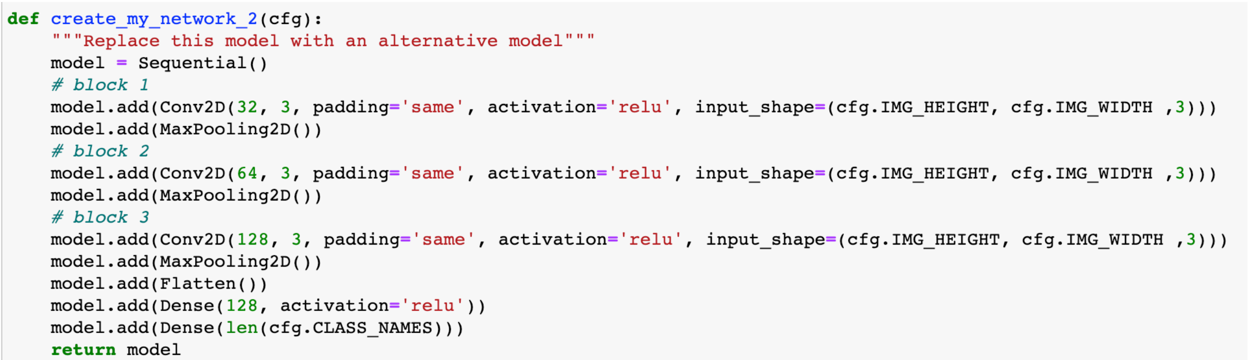


```
Label: [False False True]
> > > Training my-network-1-adam epoch 1 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.3094 - accuracy: 0.3281
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 7s 2s/step - loss: 1.3094 - accuracy: 0.3281 - val_loss: 1.0910 - val_accurac
y: 0.3359
> > > Training my-network-1-adam epoch 2 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.0564 - accuracy: 0.3906
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 6s 1s/step - loss: 1.0564 - accuracy: 0.3906 - val_loss: 0.9986 - val_accurac
y: 0.4688
> > > Training my-network-1-adam epoch 3 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.0308 - accuracy: 0.4766
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 1.0308 - accuracy: 0.4766 - val_loss: 0.9338 - val_accurac
y: 0.6562
> > > Training my-network-1-adam epoch 4 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.9003 - accuracy: 0.6172
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.9003 - accuracy: 0.6172 - val_loss: 0.8072 - val_accurac
y: 0.6562
> > > Training my-network-1-adam epoch 5 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.8394 - accuracy: 0.6328
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.8394 - accuracy: 0.6328 - val_loss: 0.7334 - val_accurac
y: 0.7109
> > > Training my-network-1-adam epoch 6 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.7502 - accuracy: 0.6562
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.7502 - accuracy: 0.6562 - val_loss: 0.8474 - val_accurac
y: 0.6172
> > > Training my-network-1-adam epoch 7 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.8185 - accuracy: 0.6250
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.8185 - accuracy: 0.6250 - val_loss: 0.6619 - val_accurac
y: 0.7500
> > > Training my-network-1-adam epoch 8 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.6572 - accuracy: 0.7656
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.6572 - accuracy: 0.7656 - val_loss: 0.6474 - val_accurac
y: 0.7188
> > > Training my-network-1-adam epoch 9 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.5769 - accuracy: 0.7891
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.5769 - accuracy: 0.7891 - val_loss: 0.5442 - val_accurac
y: 0.6953
> > > Training my-network-1-adam epoch 10 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.5044 - accuracy: 0.7734
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.5044 - accuracy: 0.7734 - val_loss: 0.4524 - val_accurac
y: 0.8047
> > > Training my-network-1-adam epoch 11 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.4549 - accuracy: 0.7812
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.4549 - accuracy: 0.7812 - val_loss: 0.3581 - val_accurac
y: 0.8359
> > > Training my-network-1-adam epoch 12 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.4199 - accuracy: 0.8359
Epoch 00001: saving model to _Temporary/my-network-1-adam/checkpoint.ckpt
4/4 [==============================] - 5s 1s/step - loss: 0.4199 - accuracy: 0.8359 - val_loss: 0.2483 - val_accurac
y: 0.9219
> > > Training of the requested number of epochs 12 done successfully.
```

2. create_my_network_2:

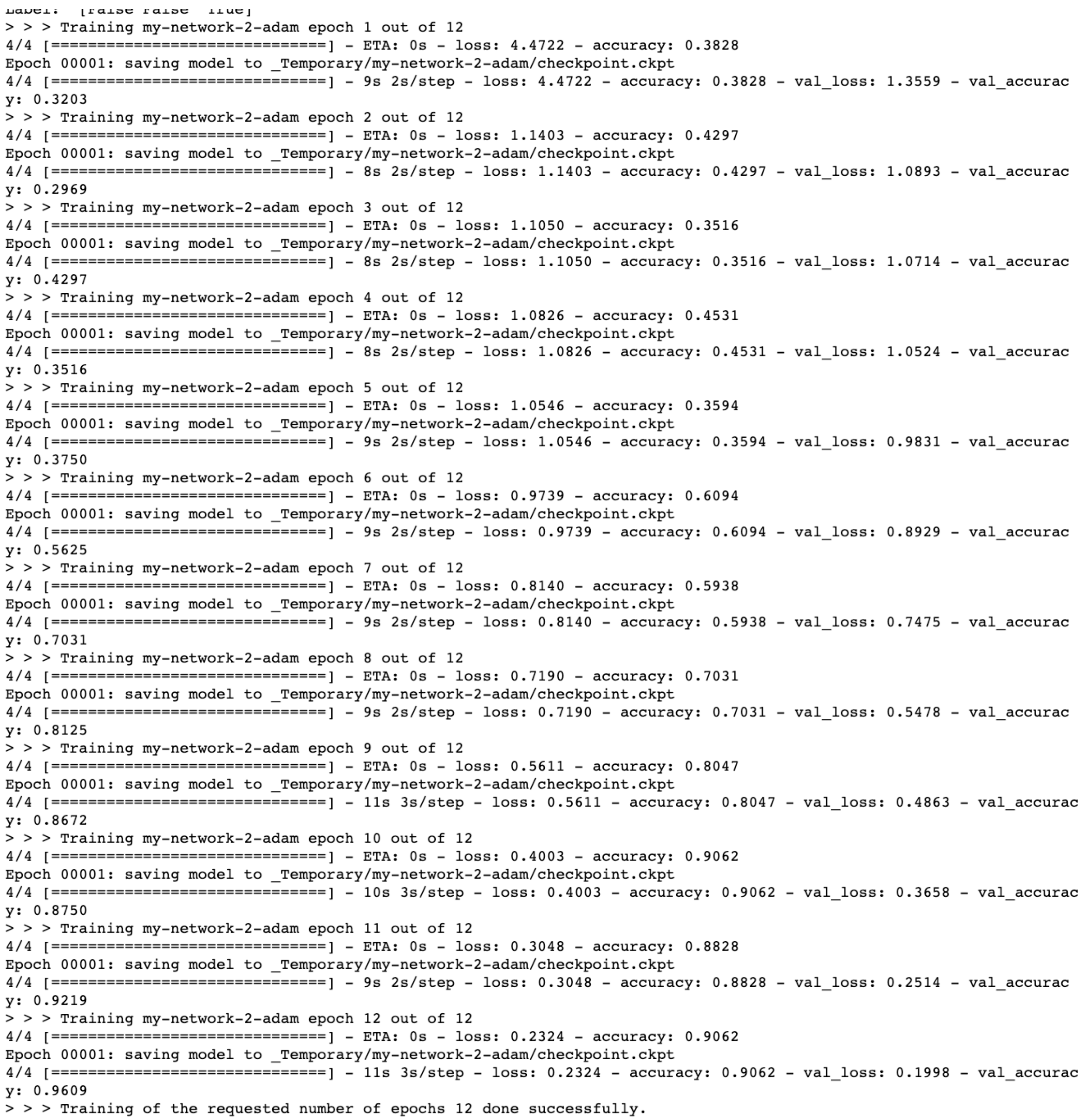
Trail #1: For this function I tried creating 4 layers at very 1st. with Conv2D 16, 32,64,128. But training went very slow and accuracy was around 30%. Which is less.
The training loss kept going up and down, while the validation accuracy, after reaching value = 0.3, did not improve at all. Conclusion: very deep networks with extreme dense (264) do not train well.

Trail #2: So, we have to reduce dense to 128 and remove 1st layer with Conv2D 16.
So, then I deleted last layer which was Conv2D 16 and dense (128) after that training went smooth and got validation accuracy **0.96.**



```
def create_my_network_2(cfg):
    """Replace this model with an alternative model"""
    model = Sequential()
    # block 1
    model.add(Conv2D(32, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 2
    model.add(Conv2D(64, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 3
    model.add(Conv2D(128, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(len(cfg.CLASS_NAMES)))
    return model
```

The output after 12 epochs:

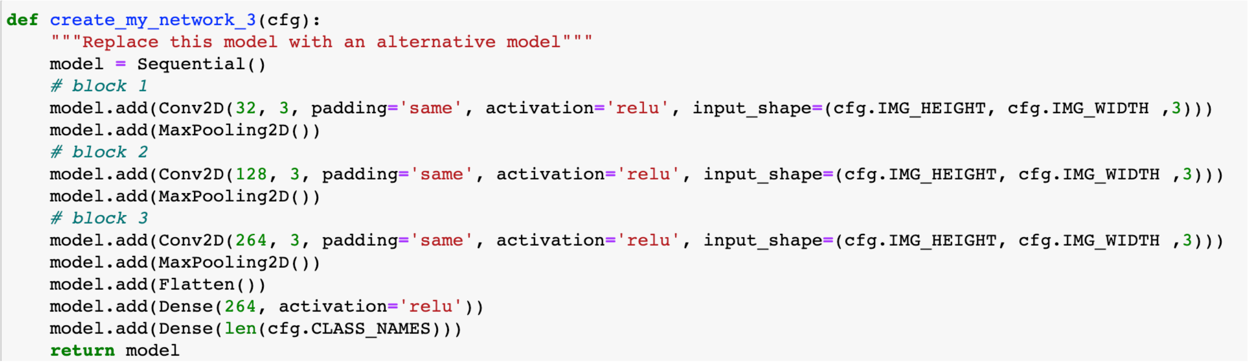


```
Label: [False False True]
> > > Training my-network-2-adam epoch 1 out of 12
4/4 [==============================] - ETA: 0s - loss: 4.4722 - accuracy: 0.3828
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 4.4722 - accuracy: 0.3828 - val_loss: 1.3559 - val_accurac
y: 0.3203
> > > Training my-network-2-adam epoch 2 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.1403 - accuracy: 0.4297
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 8s 2s/step - loss: 1.1403 - accuracy: 0.4297 - val_loss: 1.0893 - val_accurac
y: 0.2969
> > > Training my-network-2-adam epoch 3 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.1050 - accuracy: 0.3516
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 8s 2s/step - loss: 1.1050 - accuracy: 0.3516 - val_loss: 1.0714 - val_accurac
y: 0.4297
> > > Training my-network-2-adam epoch 4 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.0826 - accuracy: 0.4531
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 8s 2s/step - loss: 1.0826 - accuracy: 0.4531 - val_loss: 1.0524 - val_accurac
y: 0.3516
> > > Training my-network-2-adam epoch 5 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.0546 - accuracy: 0.3594
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 1.0546 - accuracy: 0.3594 - val_loss: 0.9831 - val_accurac
y: 0.3750
> > > Training my-network-2-adam epoch 6 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.9739 - accuracy: 0.6094
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 0.9739 - accuracy: 0.6094 - val_loss: 0.8929 - val_accurac
y: 0.5625
> > > Training my-network-2-adam epoch 7 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.8140 - accuracy: 0.5938
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 0.8140 - accuracy: 0.5938 - val_loss: 0.7475 - val_accurac
y: 0.7031
> > > Training my-network-2-adam epoch 8 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.7190 - accuracy: 0.7031
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 0.7190 - accuracy: 0.7031 - val_loss: 0.5478 - val_accurac
y: 0.8125
> > > Training my-network-2-adam epoch 9 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.5611 - accuracy: 0.8047
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 11s 3s/step - loss: 0.5611 - accuracy: 0.8047 - val_loss: 0.4863 - val_accurac
y: 0.8672
> > > Training my-network-2-adam epoch 10 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.4003 - accuracy: 0.9062
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 10s 3s/step - loss: 0.4003 - accuracy: 0.9062 - val_loss: 0.3658 - val_accurac
y: 0.8750
> > > Training my-network-2-adam epoch 11 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.3048 - accuracy: 0.8828
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 9s 2s/step - loss: 0.3048 - accuracy: 0.8828 - val_loss: 0.2514 - val_accurac
y: 0.9219
> > > Training my-network-2-adam epoch 12 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.2324 - accuracy: 0.9062
Epoch 00001: saving model to _Temporary/my-network-2-adam/checkpoint.ckpt
4/4 [==============================] - 11s 3s/step - loss: 0.2324 - accuracy: 0.9062 - val_loss: 0.1998 - val_accurac
y: 0.9609
> > > Training of the requested number of epochs 12 done successfully.
```

3. create_my_network_3:

Trail #1: For this function I tried creating 5 layers at very 1st. with Conv2D 32,128, 264. But training went very slow and accuracy was around 40%. Which is less.
The training loss kept going up and down, while the validation accuracy, after reaching value = 0.4, did not improve at all. Conclusion: very deep networks with extreme dense (10) do not train well.

Trail #2: So, we have to increase dense to 264 and remove 1st layer with Conv2D 16.
So, then I deleted last layer which was Conv2D 16 and dense (264) after that training went smooth and got validation accuracy **0.96.**

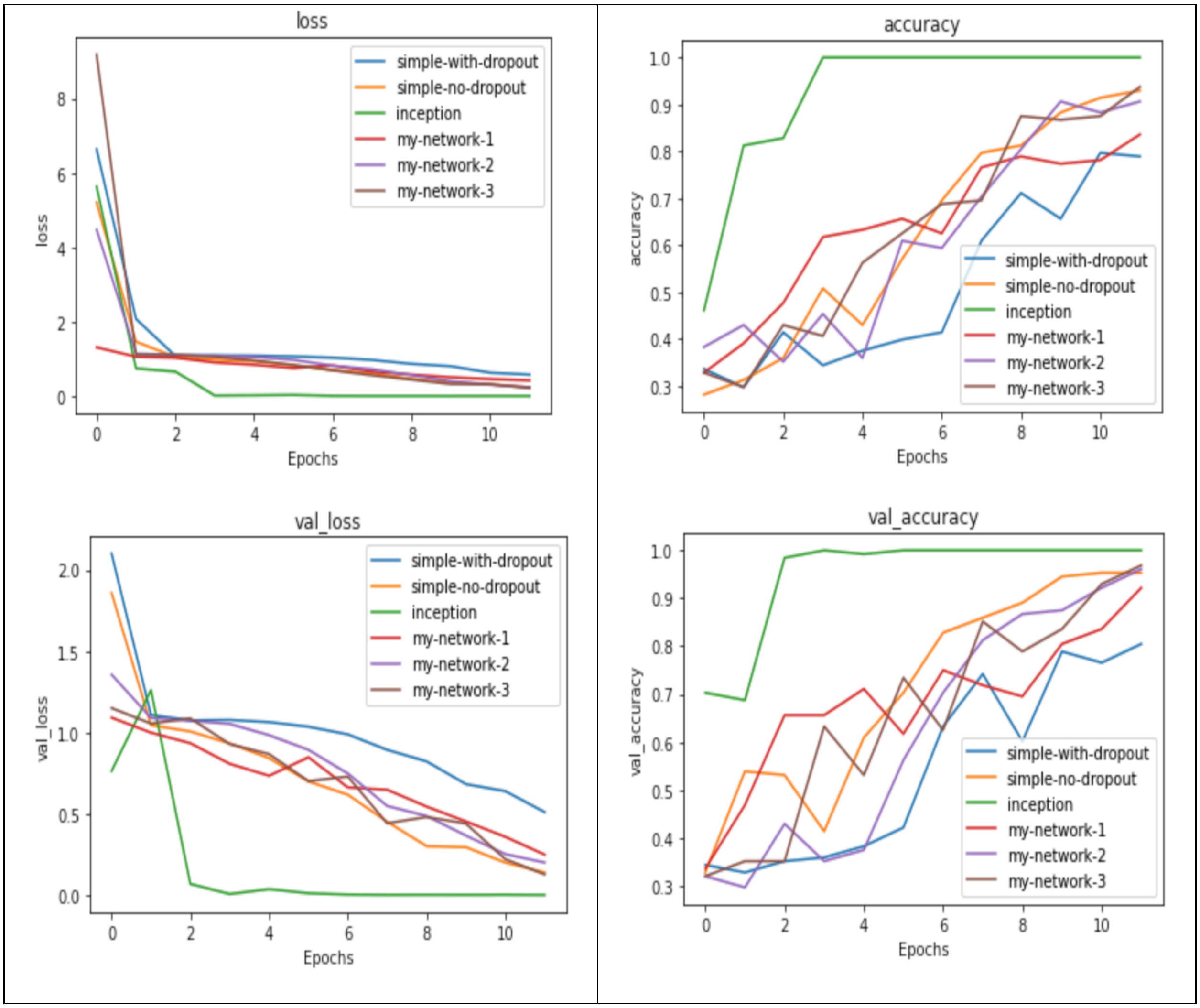


```
def create_my_network_3(cfg):
    """Replace this model with an alternative model"""
    model = Sequential()
    # block 1
    model.add(Conv2D(32, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 2
    model.add(Conv2D(128, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    # block 3
    model.add(Conv2D(264, 3, padding='same', activation='relu', input_shape=(cfg.IMG_HEIGHT, cfg.IMG_WIDTH ,3)))
    model.add(MaxPooling2D())
    model.add(Flatten())
    model.add(Dense(264, activation='relu'))
    model.add(Dense(len(cfg.CLASS_NAMES)))
    return model
```

The output for the model is as below:

```
> > > Training my-network-3-adam epoch 1 out of 12
4/4 [==============================] - ETA: 0s - loss: 9.1770 - accuracy: 0.3281
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 42s 11s/step - loss: 9.1770 - accuracy: 0.3281 - val_loss: 1.1498 - val_accura
cy: 0.3203
> > > Training my-network-3-adam epoch 2 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.1123 - accuracy: 0.2969
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 35s 9s/step - loss: 1.1123 - accuracy: 0.2969 - val_loss: 1.0547 - val_accurac
y: 0.3516
> > > Training my-network-3-adam epoch 3 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.1080 - accuracy: 0.4297
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 37s 9s/step - loss: 1.1080 - accuracy: 0.4297 - val_loss: 1.0862 - val_accurac
y: 0.3516
> > > Training my-network-3-adam epoch 4 out of 12
4/4 [==============================] - ETA: 0s - loss: 1.0584 - accuracy: 0.4062
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 45s 11s/step - loss: 1.0584 - accuracy: 0.4062 - val_loss: 0.9286 - val_accura
cy: 0.6328
> > > Training my-network-3-adam epoch 5 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.9608 - accuracy: 0.5625
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 46s 11s/step - loss: 0.9608 - accuracy: 0.5625 - val_loss: 0.8668 - val_accura
cy: 0.5312
> > > Training my-network-3-adam epoch 6 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.8335 - accuracy: 0.6250
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 29s 7s/step - loss: 0.8335 - accuracy: 0.6250 - val_loss: 0.7004 - val_accurac
y: 0.7344
> > > Training my-network-3-adam epoch 7 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.6917 - accuracy: 0.6875
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 28s 7s/step - loss: 0.6917 - accuracy: 0.6875 - val_loss: 0.7283 - val_accurac
y: 0.6250
> > > Training my-network-3-adam epoch 8 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.5560 - accuracy: 0.6953
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 36s 9s/step - loss: 0.5560 - accuracy: 0.6953 - val_loss: 0.4408 - val_accurac
y: 0.8516
> > > Training my-network-3-adam epoch 9 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.4414 - accuracy: 0.8750
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 37s 9s/step - loss: 0.4414 - accuracy: 0.8750 - val_loss: 0.4800 - val_accurac
y: 0.7891
> > > Training my-network-3-adam epoch 10 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.3207 - accuracy: 0.8672
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 39s 10s/step - loss: 0.3207 - accuracy: 0.8672 - val_loss: 0.4394 - val_accura
cy: 0.8359
> > > Training my-network-3-adam epoch 11 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.3179 - accuracy: 0.8750
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 32s 8s/step - loss: 0.3179 - accuracy: 0.8750 - val_loss: 0.2203 - val_accurac
y: 0.9297
> > > Training my-network-3-adam epoch 12 out of 12
4/4 [==============================] - ETA: 0s - loss: 0.2126 - accuracy: 0.9375
Epoch 00001: saving model to _Temporary/my-network-3-adam/checkpoint.ckpt
4/4 [==============================] - 41s 10s/step - loss: 0.2126 - accuracy: 0.9375 - val_loss: 0.1270 - val_accura
cy: 0.9688
> > > Training of the requested number of epochs 12 done successfully.
```

Final Outputs for all the Model is as below:
In the graphs loss, val_loss, accuracy and val_accuracy for all the models(my-model 1, 2, 3).


loss
loss
Epochs
simple-with-dropout
simple-no-dropout
inception
my-network-1
my-network-2
my-network-3
accuracy
accuracy
Epochs
simple-with-dropout
simple-no-dropout
inception
my-network-1
my-network-2
my-network-3
val_loss
val_loss
Epochs
simple-with-dropout
simple-no-dropout
inception
my-network-1
my-network-2
my-network-3
val_accuracy
val_accuracy
Epochs
simple-with-dropout
simple-no-dropout
inception
my-network-1
my-network-2
my-network-3