NNFL Project

November 19, 2018

1 Importing necessary libraries

In [95]:
```
import skimage.data
import skimage.transform

import os
import numpy as np
import cv2
import pickle
from imutils import paths

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from keras.models import Sequential
from keras.models import Model
from keras.optimizers import SGD
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras.layers import Concatenate
from keras.preprocessing.image import ImageDataGenerator
from keras.engine.input_layer import Input
from keras.utils.vis_utils import plot_model
from keras import backend as K

# set the matplotlib backend so figures can be saved in the background
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
matplotlib.use("Agg")
```

```
# Allow image embeding in notebook
%matplotlib inline
```

In [3]:
```
#data set paths
BTS_ROOT = '/media/prithvi/DATA/acads/NNFL/assignment 2/BelgiumTS datasets/'
BTSC_TRAINING = os.path.join(BTS_ROOT,"BelgiumTSC_Training")
BTSC_TESTING= os.path.join(BTS_ROOT,"BelgiumTSC_Testing")
BTSC_TRAINING_SMALL = os.path.join(BTS_ROOT,"BelgiumTSC_Training_small")
```

2 Loading the Dataset

In [90]:
```
#function to load data from data_dir
def load_data(data_dir):
    # Get all subdirectories of data_dir. Each represents a label.
    directories = [d for d in os.listdir(data_dir)
                   if os.path.isdir(os.path.join(data_dir, d))]

    # Loop through the label directories and collect the data in
    # two lists, labels and images.
    labels = []
    images = []
    for d in directories:
        label_dir = os.path.join(data_dir, d)
        file_names = [os.path.join(label_dir, f)
                      for f in os.listdir(label_dir)
                      if f.endswith(".ppm")]
        for f in file_names:
            images.append(skimage.data.imread(f))
            labels.append(int(d))
    return images, labels
```

In [92]:
```
#loading the data
# images, labels = load_data(BTSC_TRAINING)
images, labels = load_data(BTSC_TRAINING)
```

2.1 Displaying an example image from each class

In [93]:
```
def display_images_and_labels(images, labels):
    """Display the first image of each label."""
    unique_labels = set(labels)
    plt.figure(figsize=(15, 15))
    i = 1
    for label in unique_labels:
        # Pick the first image for each label.
        image = images[labels.index(label)]
        plt.subplot(8, 8, i)  # A grid of 8 rows x 8 columns
        plt.axis('off')
        plt.title("Label {0} ({1})".format(label, labels.count(label)))
```

```python
            i += 1
            _ = plt.imshow(image)
        plt.show()

    display_images_and_labels(images, labels)
```

2.2 Resizing Images and One-Hot encoding class labels

In [7]:
```python
'''
resizing, training data
storing it as np array of floats normalised to interval [0,1]
'''
trainX = np.array([ cv2.resize(image,(48,48)) for image in images ] , dtype="float")/25
```

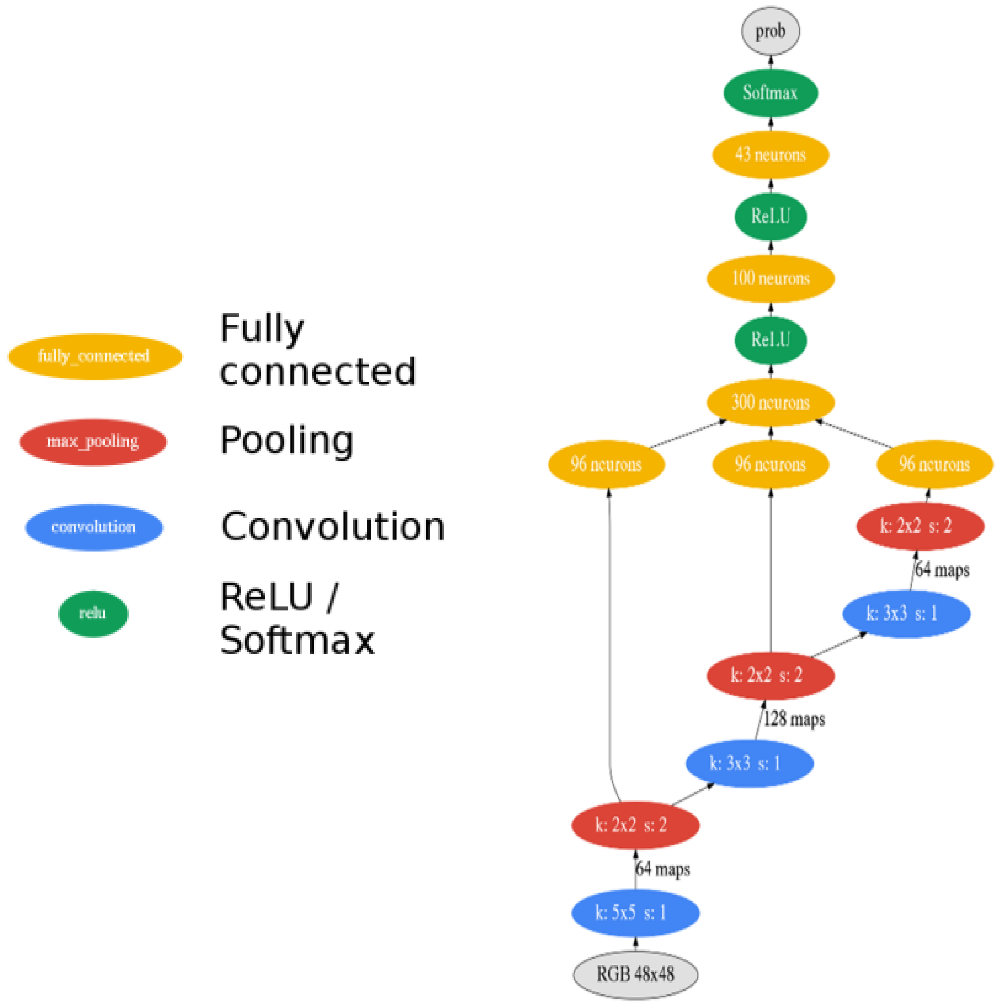
```
#converting labels to one hot encoding
lb= LabelBinarizer()
trainY = lb.fit_transform(labels) #finds unique labels and one hot encodes

#reading test data
testims,testLabels= load_data(BTSC_TESTING)
testX = np.array([ cv2.resize(image,(48,48)) for image in testims], dtype="float")/255
testY = lb.transform(testLabels) #transform() just encodes based on classes detected i
```

In [8]:
```
print("No. of training examples = ", len(trainX))
print(trainX[0].shape)
print("No. of test examples = ",len(testX))
print("No. of classes =",len(lb.classes_))
```

```
No. of training examples =  4575
(48, 48, 3)
No. of test examples =  2554
No. of classes = 62
```

3 Architecture

In [104]:
```
model_from_paper = mpimg.imread("architecture.png")
legend =  skimage.data.imread("legend.png")
plt.figure(figsize=(10,10),dpi=512)
plt.subplot(1,2,1)
plt.imshow(legend,aspect='equal')
plt.axis("off");
plt.subplot(1,2,2)
plt.imshow(model_from_paper,aspect='auto')
plt.axis('off');
```



4 Making the Network

4.1 Creating Layers and compiling the model

```
In [114]: #inpute layer
          ip = Input(shape=trainX[0].shape)

          #first convolutional layer
          c1 = Conv2D(filters=64, kernel_size=(5,5), strides=1, padding="same")(ip)

          #first pooling layer
          p1 = MaxPooling2D(pool_size = (2,2), strides=2)(c1)
          p1 = Dropout(0.25)(p1)
```

```python
#second convolutional layer
c2 = Conv2D(filters=128, kernel_size=(3,3), strides=1, padding="same")(p1)

#second pooling layer
p2 = MaxPooling2D(pool_size = (2,2), strides=2)(c2)
p2 = Dropout(0.25)(p2)

#third convolutional layer
c3 = Conv2D(filters=64, kernel_size=(3,3), strides=1, padding="same")(p2)

#third pooling layer
p3 = MaxPooling2D(pool_size = (2,2), strides=2)(c3)
p3 = Dropout(0.25)(p3)

#flattening outputs of p1,p2 and p3
fl1 = Flatten()(p1)
fl2 = Flatten()(p2)
fl3 = Flatten()(p3)

#fully connected layers
fc1 = Dense(96)(fl1)
fc1 = Dropout(0.25)(fc1)

fc2 = Dense(96)(fl2)
fc2 = Dropout(0.25)(fc2)

fc3 = Dense(96)(fl3)
fc3 = Dropout(0.25)(fc3)

#merging layer
m1 = Concatenate()([fc1, fc2, fc3])

#fully connected layer
fc4 = Dense(300, activation = 'relu')(m1)
fc4 = Dropout(0.25)(fc4)

#fully connected
fc5 = Dense(100,activation = 'relu')(fc4)
fc5 = Dropout(0.25)(fc5)

#fully connected (final layer - 62 outputs)
fc6 = Dense(len(lb.classes_), activation = 'softmax')(fc5)


#model
model = Model(inputs = ip, outputs = fc6)

# initialize our initial learning rate and # of epochs to train for
```

```
INIT_LR = 0.03
MOMENTUM = 0.3

#INIT_LR = 0.03 and MOMENTUM = 0.3 are the best settings found so far

# compile the model using SGD as our optimizer and categorical
# cross-entropy loss (softmax loss)
opt = SGD(lr = INIT_LR, momentum = MOMENTUM)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])
```

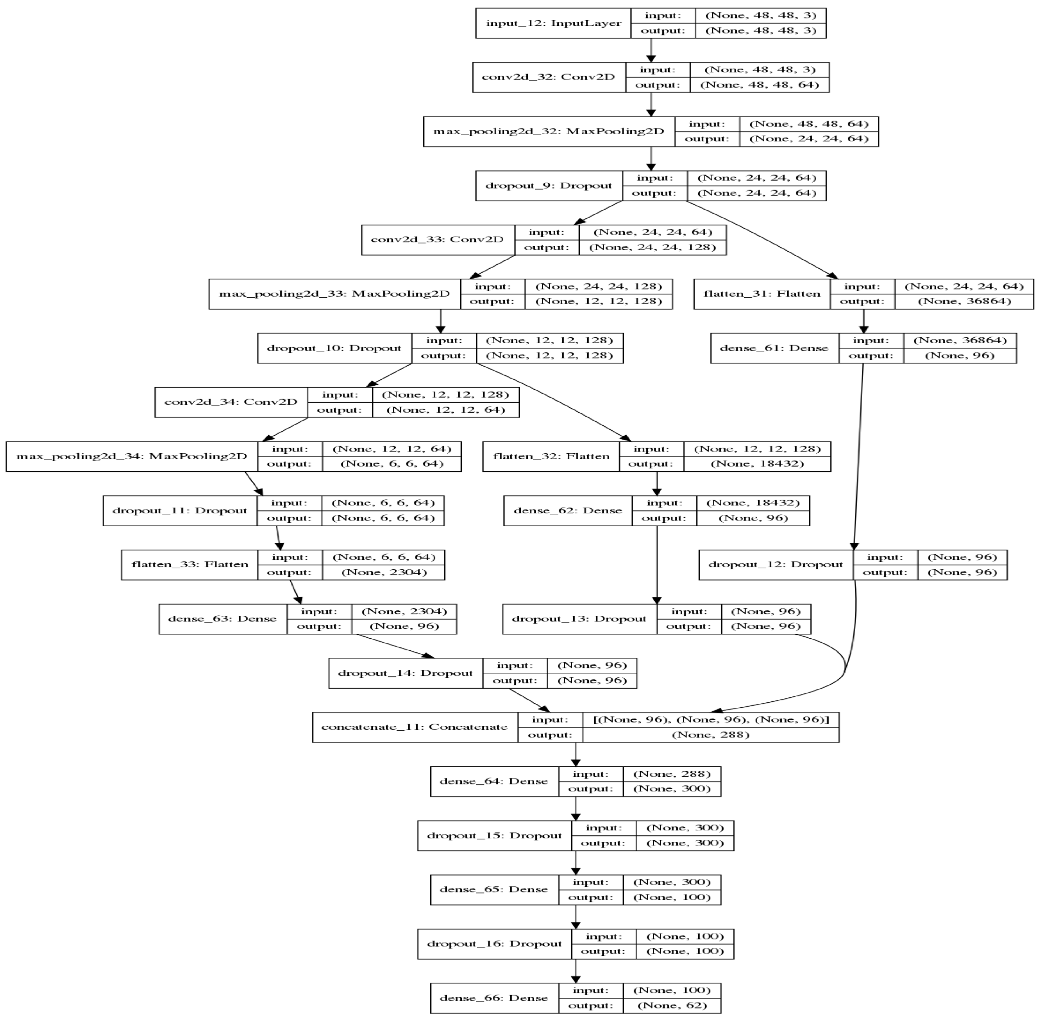
In [115]:
```
print(model.summary())
plot_model(model, show_shapes= True)
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_12 (InputLayer)           (None, 48, 48, 3)    0
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 48, 48, 64)   4864        input_12[0][0]
__________________________________________________________________________________________________
max_pooling2d_32 (MaxPooling2D) (None, 24, 24, 64)   0           conv2d_32[0][0]
__________________________________________________________________________________________________
dropout_9 (Dropout)             (None, 24, 24, 64)   0           max_pooling2d_32[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 24, 24, 128)  73856       dropout_9[0][0]
__________________________________________________________________________________________________
max_pooling2d_33 (MaxPooling2D) (None, 12, 12, 128)  0           conv2d_33[0][0]
__________________________________________________________________________________________________
dropout_10 (Dropout)            (None, 12, 12, 128)  0           max_pooling2d_33[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 12, 12, 64)   73792       dropout_10[0][0]
__________________________________________________________________________________________________
max_pooling2d_34 (MaxPooling2D) (None, 6, 6, 64)     0           conv2d_34[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 6, 6, 64)     0           max_pooling2d_34[0][0]
__________________________________________________________________________________________________
flatten_31 (Flatten)            (None, 36864)        0           dropout_9[0][0]
__________________________________________________________________________________________________
flatten_32 (Flatten)            (None, 18432)        0           dropout_10[0][0]
__________________________________________________________________________________________________
flatten_33 (Flatten)            (None, 2304)         0           dropout_11[0][0]
__________________________________________________________________________________________________
dense_61 (Dense)                (None, 96)           3539040     flatten_31[0][0]
__________________________________________________________________________________________________
dense_62 (Dense)                (None, 96)           1769568     flatten_32[0][0]
__________________________________________________________________________________________________
dense_63 (Dense)                (None, 96)           221280      flatten_33[0][0]
__________________________________________________________________________________________________
```

```
dropout_12 (Dropout)            (None, 96)           0           dense_61[0][0]
__________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 96)           0           dense_62[0][0]
__________________________________________________________________________________________________
dropout_14 (Dropout)            (None, 96)           0           dense_63[0][0]
__________________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 288)          0           dropout_12[0][0]
                                                                 dropout_13[0][0]
                                                                 dropout_14[0][0]
__________________________________________________________________________________________________
dense_64 (Dense)                (None, 300)          86700       concatenate_11[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 300)          0           dense_64[0][0]
__________________________________________________________________________________________________
dense_65 (Dense)                (None, 100)          30100       dropout_15[0][0]
__________________________________________________________________________________________________
dropout_16 (Dropout)            (None, 100)          0           dense_65[0][0]
__________________________________________________________________________________________________
dense_66 (Dense)                (None, 62)           6262        dropout_16[0][0]
==================================================================================================
Total params: 5,805,462
Trainable params: 5,805,462
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

4.2 Architecture of the network made

```
In [116]: model_diagram=skimage.data.imread("model.png")
          plt.figure(figsize=(10,10),dpi=512)
          plt.imshow(model_diagram,aspect='auto')
          plt.axis("off");
```



5 Training

```
In [119]: #train the model
          EPOCHS = 50
          BATCH_SIZE = 32

          gen = ImageDataGenerator(width_shift_range = 0.1, height_shift_range = 0.1, zoom_rang
          H = model.fit_generator(gen.flow(trainX,trainY), validation_data=(testX,testY), steps

Epoch 1/50
142/142 [==============================] - 49s 345ms/step - loss: 0.3526 - acc: 0.8875 - val_lo
Epoch 2/50
142/142 [==============================] - 50s 350ms/step - loss: 0.3829 - acc: 0.8818 - val_lo
Epoch 3/50
```

```
142/142 [==============================] - 49s 346ms/step - loss: 0.3714 - acc: 0.8840 - val_lo
Epoch 4/50
142/142 [==============================] - 49s 346ms/step - loss: 0.3412 - acc: 0.8886 - val_lo
Epoch 5/50
142/142 [==============================] - 52s 364ms/step - loss: 0.3184 - acc: 0.8965 - val_lo
Epoch 6/50
142/142 [==============================] - 52s 367ms/step - loss: 0.3229 - acc: 0.8961 - val_lo
Epoch 7/50
142/142 [==============================] - 49s 344ms/step - loss: 0.3218 - acc: 0.8998 - val_lo
Epoch 8/50
142/142 [==============================] - 52s 364ms/step - loss: 0.3202 - acc: 0.8994 - val_lo
Epoch 9/50
142/142 [==============================] - 48s 342ms/step - loss: 0.3202 - acc: 0.8998 - val_lo
Epoch 10/50
142/142 [==============================] - 49s 342ms/step - loss: 0.2935 - acc: 0.9087 - val_lo
Epoch 11/50
142/142 [==============================] - 49s 347ms/step - loss: 0.2887 - acc: 0.9109 - val_lo
Epoch 12/50
142/142 [==============================] - 50s 353ms/step - loss: 0.2899 - acc: 0.9062 - val_lo
Epoch 13/50
142/142 [==============================] - 51s 358ms/step - loss: 0.2582 - acc: 0.9170 - val_lo
Epoch 14/50
142/142 [==============================] - 49s 345ms/step - loss: 0.2752 - acc: 0.9135 - val_lo
Epoch 15/50
142/142 [==============================] - 49s 346ms/step - loss: 0.2796 - acc: 0.9155 - val_lo
Epoch 16/50
142/142 [==============================] - 49s 346ms/step - loss: 0.2620 - acc: 0.9183 - val_lo
Epoch 17/50
142/142 [==============================] - 49s 346ms/step - loss: 0.2381 - acc: 0.9309 - val_lo
Epoch 18/50
142/142 [==============================] - 49s 347ms/step - loss: 0.2468 - acc: 0.9223 - val_lo
Epoch 19/50
142/142 [==============================] - 49s 347ms/step - loss: 0.2344 - acc: 0.9258 - val_lo
Epoch 20/50
142/142 [==============================] - 54s 382ms/step - loss: 0.2228 - acc: 0.9289 - val_lo
Epoch 21/50
142/142 [==============================] - 50s 350ms/step - loss: 0.2194 - acc: 0.9238 - val_lo
Epoch 22/50
142/142 [==============================] - 49s 347ms/step - loss: 0.2386 - acc: 0.9252 - val_lo
Epoch 23/50
142/142 [==============================] - 50s 349ms/step - loss: 0.2170 - acc: 0.9344 - val_lo
Epoch 24/50
142/142 [==============================] - 51s 358ms/step - loss: 0.2308 - acc: 0.9276 - val_lo
Epoch 25/50
142/142 [==============================] - 50s 349ms/step - loss: 0.2321 - acc: 0.9269 - val_lo
Epoch 26/50
142/142 [==============================] - 49s 346ms/step - loss: 0.2009 - acc: 0.9351 - val_lo
Epoch 27/50
```

```
142/142 [==============================] - 49s 346ms/step - loss: 0.2120 - acc: 0.9344 - val_lo
Epoch 28/50
142/142 [==============================] - 49s 348ms/step - loss: 0.2283 - acc: 0.9300 - val_lo
Epoch 29/50
142/142 [==============================] - 49s 347ms/step - loss: 0.2115 - acc: 0.9333 - val_lo
Epoch 30/50
142/142 [==============================] - 50s 353ms/step - loss: 0.1902 - acc: 0.9428 - val_lo
Epoch 31/50
142/142 [==============================] - 50s 352ms/step - loss: 0.1892 - acc: 0.9414 - val_lo
Epoch 32/50
142/142 [==============================] - 50s 349ms/step - loss: 0.1887 - acc: 0.9428 - val_lo
Epoch 33/50
142/142 [==============================] - 50s 350ms/step - loss: 0.1837 - acc: 0.9410 - val_lo
Epoch 34/50
142/142 [==============================] - 49s 344ms/step - loss: 0.1841 - acc: 0.9397 - val_lo
Epoch 35/50
142/142 [==============================] - 50s 349ms/step - loss: 0.2043 - acc: 0.9423 - val_lo
Epoch 36/50
142/142 [==============================] - 51s 356ms/step - loss: 0.1529 - acc: 0.9555 - val_lo
Epoch 37/50
142/142 [==============================] - 49s 347ms/step - loss: 0.1718 - acc: 0.9472 - val_lo
Epoch 38/50
142/142 [==============================] - 49s 348ms/step - loss: 0.1933 - acc: 0.9401 - val_lo
Epoch 39/50
142/142 [==============================] - 50s 354ms/step - loss: 0.1804 - acc: 0.9467 - val_lo
Epoch 40/50
142/142 [==============================] - 53s 370ms/step - loss: 0.1772 - acc: 0.9434 - val_lo
Epoch 41/50
142/142 [==============================] - 53s 375ms/step - loss: 0.1841 - acc: 0.9417 - val_lo
Epoch 42/50
142/142 [==============================] - 52s 367ms/step - loss: 0.1771 - acc: 0.9448 - val_lo
Epoch 43/50
142/142 [==============================] - 52s 367ms/step - loss: 0.1640 - acc: 0.9478 - val_lo
Epoch 44/50
142/142 [==============================] - 53s 371ms/step - loss: 0.1641 - acc: 0.9503 - val_lo
Epoch 45/50
142/142 [==============================] - 53s 377ms/step - loss: 0.1715 - acc: 0.9465 - val_lo
Epoch 46/50
142/142 [==============================] - 54s 379ms/step - loss: 0.1729 - acc: 0.9474 - val_lo
Epoch 47/50
142/142 [==============================] - 52s 369ms/step - loss: 0.1724 - acc: 0.9459 - val_lo
Epoch 48/50
142/142 [==============================] - 53s 370ms/step - loss: 0.1611 - acc: 0.9520 - val_lo
Epoch 49/50
142/142 [==============================] - 51s 361ms/step - loss: 0.1539 - acc: 0.9527 - val_lo
Epoch 50/50
142/142 [==============================] - 49s 345ms/step - loss: 0.1489 - acc: 0.9529 - val_lo
```

```
In [122]: # save the model and label binarizer to disk
          model.save("model_dropout25")
          f = open("binarizer", "wb")
          f.write(pickle.dumps(lb))
          f.close()
```

6 Testing

```
In [121]: # evaluate the network
          predictions = model.predict(testX, batch_size=32)
          print(classification_report(testY.argmax(axis=1),predictions.argmax(axis=1), labels=
```

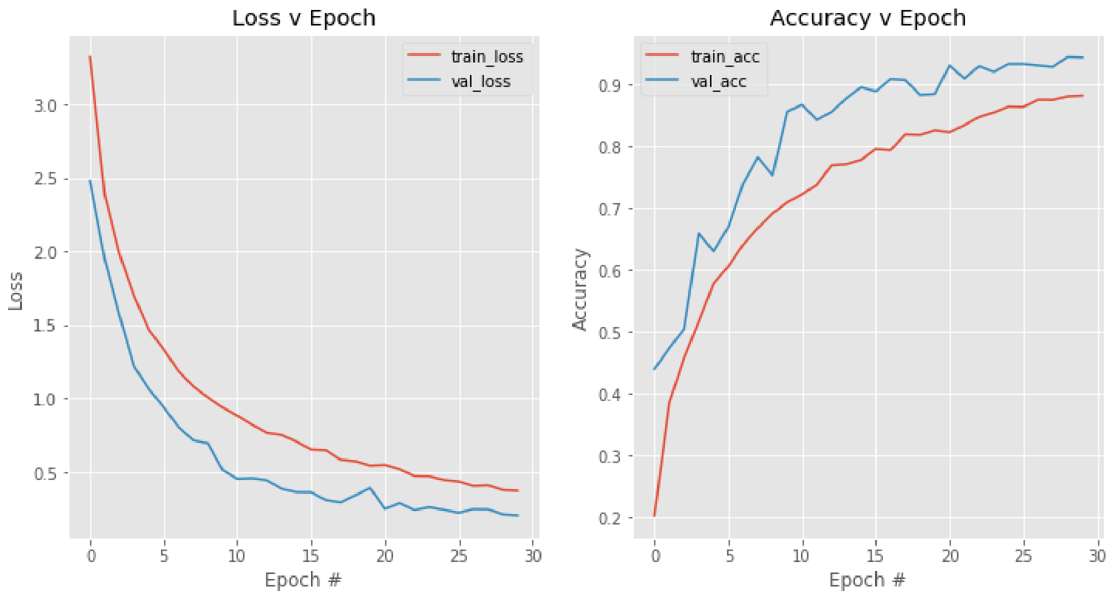
```
        precision    recall  f1-score   support

 0          0.86      1.00      0.92         6
 1          0.96      1.00      0.98        27
 2          1.00      0.71      0.83         7
 3          0.75      0.50      0.60         6
 4          0.92      0.92      0.92        12
 5          1.00      0.67      0.80         3
 6          0.62      0.83      0.71         6
 7          0.99      1.00      0.99        90
 8          0.63      1.00      0.77        12
 9          1.00      1.00      1.00         7
10          1.00      0.96      0.98        28
11          1.00      1.00      1.00         4
12          1.00      0.67      0.80         3
13          1.00      0.90      0.95        39
14          0.62      1.00      0.77        15
15          1.00      1.00      1.00         5
16          1.00      0.25      0.40        12
17          0.98      0.99      0.99       183
18          1.00      0.98      0.99       122
19          1.00      1.00      1.00       163
20          1.00      1.00      1.00         3
21          0.98      0.98      0.98        45
22          0.98      1.00      0.99        61
23          1.00      0.80      0.89        15
24          0.91      0.77      0.83        13
25          0.60      1.00      0.75         3
26          0.50      1.00      0.67         2
27          1.00      1.00      1.00         9
28          0.91      1.00      0.95        51
29          0.93      1.00      0.97        28
30          0.97      1.00      0.99        37
31          1.00      1.00      1.00        86
32          1.00      0.99      1.00       422
```

```
          33       1.00      1.00      1.00         3
          34       1.00      1.00      1.00         9
          35       1.00      0.95      0.97       154
          36       1.00      1.00      1.00         4
          37       1.00      1.00      1.00        31
          38       0.97      1.00      0.99       213
          39       1.00      0.99      0.99        99
          40       1.00      0.96      0.98        48
          41       1.00      1.00      1.00        11
          42       0.64      1.00      0.78         9
          43       1.00      0.17      0.29         6
          44       1.00      1.00      1.00         3
          45       0.96      0.92      0.94        84
          46       0.43      0.50      0.46         6
          47       0.86      1.00      0.93        31
          48       1.00      1.00      1.00         3
          49       1.00      0.33      0.50         3
          50       1.00      1.00      1.00         3
          51       1.00      1.00      1.00         3
          52       0.75      1.00      0.86         3
          53       1.00      1.00      1.00        24
          54       1.00      1.00      1.00        48
          55       1.00      0.93      0.97        15
          56       0.97      1.00      0.99        33
          57       1.00      0.78      0.88        41
          58       0.90      1.00      0.95         9
          59       0.68      1.00      0.81        17
          60       1.00      0.55      0.71        11
          61       0.95      1.00      0.98       105

   micro avg       0.97      0.97      0.97      2554
   macro avg       0.92      0.90      0.89      2554
weighted avg       0.97      0.97      0.97      2554
```

6.1 Plotting traing curves

In [118]:
```
# plot the training loss and accuracy
N = np.arange(0, EPOCHS)
plt.style.use("ggplot")
plt.figure(figsize=(12,6))
plt.title("Training Loss and Accuracy (Simple NN)")
plt.subplot(1,2,1)
plt.title("Loss v Epoch")
plt.plot(N, H.history["loss"], label="train_loss")
plt.plot(N, H.history["val_loss"], label="val_loss")
plt.xlabel("Epoch #")
```

```
plt.ylabel("Loss")
plt.legend()
plt.subplot(1,2,2)
plt.title("Accuracy v Epoch")
plt.plot(N, H.history["acc"], label="train_acc")
plt.plot(N, H.history["val_acc"], label="val_acc")
plt.xlabel("Epoch #")
plt.ylabel("Accuracy")
plt.legend()
plt.savefig(".")
```



In [ ]: