Final Project Submission

Please fill out:

- Student name: Justin Grisanti
- Student pace: self-paced
- Scheduled project review date/time: 2/24/2022 @ TBD
- Instructor name: Claude Fried
- Blog post URL: TBD

Section 1: Business Understanding

The purpose of this section is to define the business problem and understand the stakeholders for the work that I am performing. Mount Sinai is a hospital network in New York City. The Health System includes more than 6,600 primary and specialty care physicians and 13 ambulatory surgical centers.

For years, doctors at Mount Sinai review chest x-rays and try to determine whether there a patient has pneumonia. Pneumonia is an infection of the lung. The lungs fill with fluid and make breathing difficult. Pneumonia disproportionately affects the young, the elderly, and the immunocompromised. It preys on weakness and vulnerability.

According to a 2010 study performed by the Henry Ford Health System, Pneumonia ranks second to congestive heart failure as the reason for readmission within 30 days of a previous hospitalization. Here are some important findings from the study:

- 72 percent of patients were misdiagnosed with pneumonia upon readmission to the same hospital.
- African-Americans were twice more likely than Caucasians to be misdiagnosed with pneumonia.
- Patients who smoke or have lung disease were likely to be misdiagnosed with pneumonia.
- 72 percent of the misdiagnoses occurred in the Emergency Department.
- Fewer than 33 percent of patients had any outpatient follow-up care prior to their readmission.

With these statistics in mind, Mount Sinai has contracted me to use deep learning to more accurately predict whether a patient has pneumonia, given a patient's chest x-ray.

The stakeholders of this project are patients, doctors, and radiologists.

The main purpose of this classification model is predictive, meaning that given a picture of a patients chest, the model should be able to predict whether that patient has pneumonia or not.

Section 2: Data Understanding

The data downloaded has three different folders, train data, test data, and validation data. Across all of the folders, we have 5,856 chest x-rays for patients that have/don't have pneumonia. This data is useful/appropriate for solving our business problem. Let's import our packages and then begin analyzing our train dataset.

In [2]:

```
# Import necessary libraries

import pandas as pd
from numpy.random import seed
seed(123)
```

```
import numpy as np
import random
import shutil
import math
import statistics as stat
import os
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import keras
import sklearn as sk
from sklearn import metrics
from sklearn.metrics import accuracy_score, f1_score, precision_s
from sklearn.model_selection import KFold
from keras import models, layers, regularizers, metrics, backend
from keras.models import Sequential, load_model
from keras.layers import Dense
from keras.optimizers import SGD
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from keras.preprocessing.image import ImageDataGenerator, array_t
from keras.callbacks import EarlyStopping, ModelCheckpoint
import pprint
```

Let's begin with analyzing our different chest x-ray images for our train, test and validation datasets. As we can see below, the lungs in our class 'normal' x-rays appear to be clearer than the lungs in our 'pneumonia' class, and it seems like it harder to see the heart as a result of this. However, because I am not a doctor, if I removed the labels and had to guess if a patient had pneumonia or not, I would probably do a fairly poor job. This is where deep learning comes in. We can use deep learning to analyze these images further to see if we are able to better predict whether a patient has pneumonia or not.

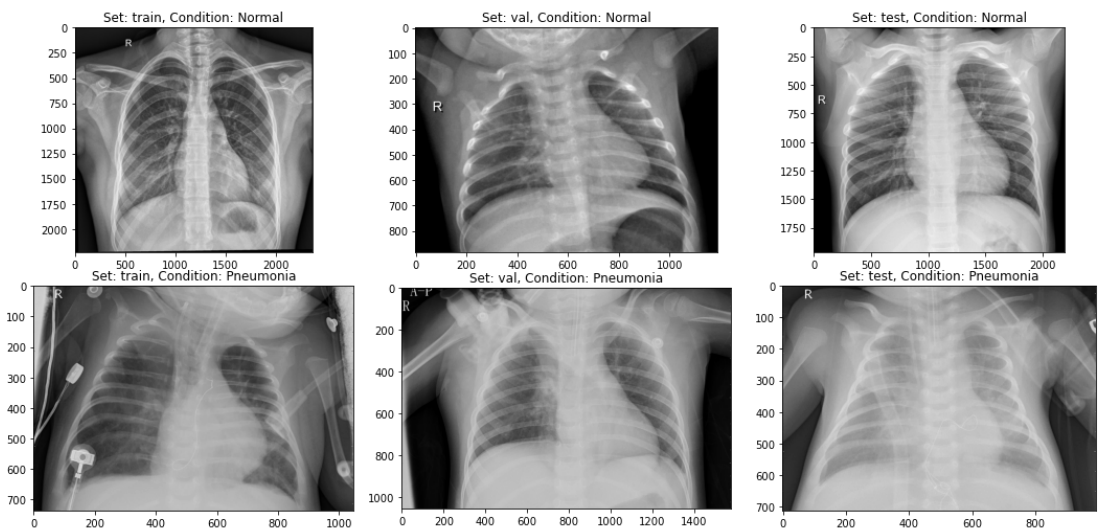
In [24]:

```
# Set important file paths, create variables for use in other are

input_path = 'data/chest_xray/'

class_labels=['NORMAL','PNEUMONIA']
X=[]
Y=[]

train_path=input_path+'/train/'
validation_path=input_path+'/val/'
test_path=input_path+'/test/'
```

In [25]:

```
# Plot a sample from each folder, showing normal x-rays vs pneumo
```

```
fig, ax = plt.subplots(2, 3, figsize=(15, 7))
ax = ax.ravel()
plt.tight_layout()

for i, _set in enumerate(['train', 'val', 'test']):
    set_path = input_path+_set
    ax[i].imshow(plt.imread(set_path+'/NORMAL/'+os.listdir(set_pa
    ax[i].set_title('Set: {}, Condition: Normal'.format(_set))
    ax[i+3].imshow(plt.imread(set_path+'/PNEUMONIA/'+os.listdir(s
    ax[i+3].set_title('Set: {}, Condition: Pneumonia'.format(_set
plt.savefig('Visualizations/chestxrays.png', bbox_inches = 'tight
```



Now, I will analyze our population:

In [6]:

```
# Hashing this code, as the initial population has been changed

#total_image_count = []
#ill_image_count = []
#normal_image_count = []

#for _set in ['train','val', 'test']:
#    total_image_count.append(len(os.listdir(input_path + _set +
#    total_image_count.append(len(os.listdir(input_path + _set +
#    normal_image_count.append(len(os.listdir(input_path + _set +
#    ill_image_count.append(len(os.listdir(input_path + _set + '/
#    print(_set + '/NORMAL:', len(os.listdir(input_path + _set +
#    print(_set + '/PNEUMONIA:',len(os.listdir(input_path + _set

#print('\nValue Counts: \nNormal:',sum(normal_image_count)/sum(to
#print('\nPopulation Breakdown: \nTrain:', (len(os.listdir(input_

#print('\nSum: ',sum(total_image_count))
```

train/NORMAL: 1341

train/PNEUMONIA: 3875

val/NORMAL: 8
val/PNEUMONIA: 8
test/NORMAL: 234
test/PNEUMONIA: 390

Value Counts:
Normal: 0.2703210382513661
Pneumonia: 0.7296789617486339

Population Breakdown:
Train: 0.8907103825136612
Test: 0.10655737704918032
Val: 0.00273224043715847

Sum: 5856

Above, we can see our total population is broken down as 27% normal chest x-rays, and 73% pneumonia chest x-rays. If our model were to randomly guess all class 1, a 73% accuracy rate would be expected. We should expect a good model to have an accuracy rating that is better than 73%. Also, our train, test and val population is split by 89.1%, 10.7% and .2%, respectively. I am going to use K-Fold Cross Validation on the train dataset. To even our population, I want to move some data to our test folder for validation when we fit the model, and also want to move data to our validation folder for our final model.

In [6]:

```
# To move images between folders

def move_img(input_source,input_dest,weight):
    files = os.listdir(input_source)
    num_files=int(len(os.listdir(input_source))*weight)
    for file_name in random.sample(files, num_files):
        shutil.move(os.path.join(input_source, file_name), input_
```

In [7]:

```
# I move 2% of our images to our validation folders

# move_img(train_path+'/NORMAL',validation_path+'NORMAL',.02)
# move_img(train_path+'/PNEUMONIA',validation_path+'/PNEUMONIA',.
```

In [17]:
```
# I move all of our of our images to our train folders

# move_img(test_path+'/NORMAL',train_path+'NORMAL',1)
# move_img(test_path+'/PNEUMONIA',train_path+'/PNEUMONIA',1)
```

In [18]:
```
# I randomly move 20% of our images to our test folders

# move_img(train_path+'/NORMAL',test_path+'NORMAL',.2)
# move_img(train_path+'/PNEUMONIA',test_path+'/PNEUMONIA',.2)
```

In [45]:
```
# Check population information

total_image_count = []
ill_image_count = []
normal_image_count = []

for _set in ['train','val', 'test']:
    total_image_count.append(len(os.listdir(input_path + _set + '
    total_image_count.append(len(os.listdir(input_path + _set + '
    normal_image_count.append(len(os.listdir(input_path + _set +
    ill_image_count.append(len(os.listdir(input_path + _set + '/P
    print(_set + '/NORMAL:', len(os.listdir(input_path + _set + '
    print(_set + '/PNEUMONIA:',len(os.listdir(input_path + _set +

print('\nValue Counts: \nNormal:',sum(normal_image_count)/sum(tot
print('\nPopulation Breakdown: \nTrain:', (len(os.listdir(input_p

print('\nSum: ',sum(total_image_count))
```

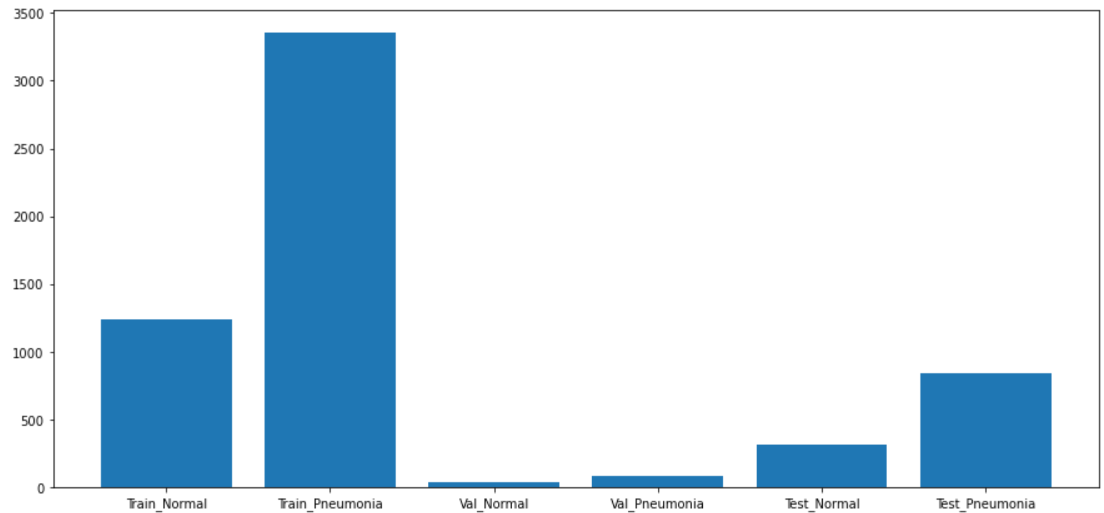
```
train/NORMAL: 1240
train/PNEUMONIA: 3351
val/NORMAL: 34
val/PNEUMONIA: 85
test/NORMAL: 309
test/PNEUMONIA: 837

Value Counts:
Normal: 0.2703210382513661
Pneumonia: 0.7296789617486339

Population Breakdown:
Train: 0.7839822404371585
Test: 0.19569672131147542
Val: 0.020321038251366612

Sum:  5856
```

In [27]:
```
# Combine lists for graphing purposes
```

```
population = [sum(normal_image_count),sum(ill_image_count)]
```

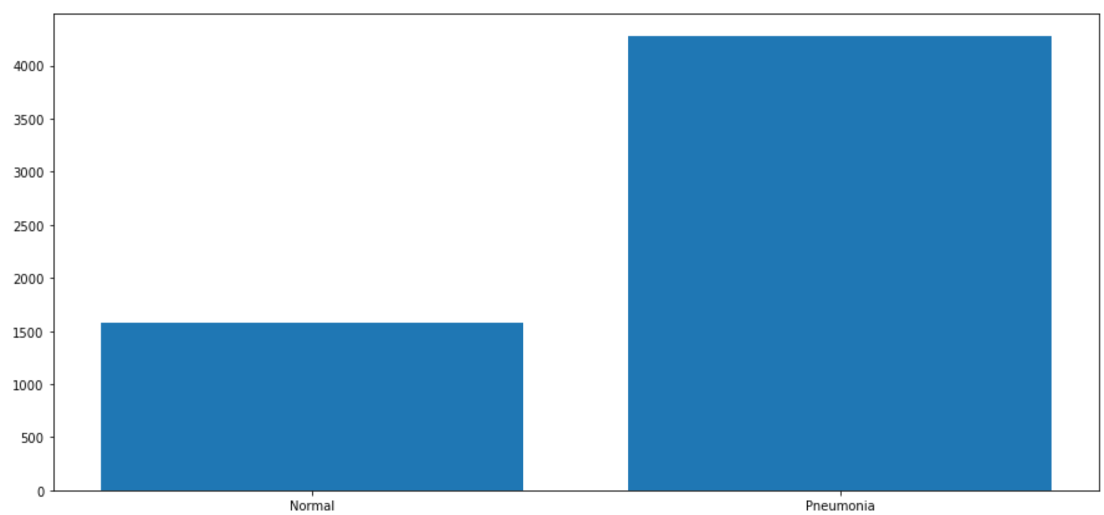
In [28]:

```
# Plot populations by folder

labels = ['Train_Normal','Train_Pneumonia','Val_Normal','Val_Pneu
fig, ax = plt.subplots(figsize=(15, 7))
ax.bar(x=labels, height=total_image_count)
plt.savefig('Visualizations/FolderPopulation.png', bbox_inches =
```



In [29]:

```
# Plot population by class

labels_2 = ['Normal','Pneumonia']
fig, ax = plt.subplots(figsize=(15, 7))
ax.bar(x=labels_2, height=population)
plt.savefig('Visualizations/ClassPopulation.png', bbox_inches = '
```



Section 3: Data Preparation

Now, I am going to prepare each image by converting the picture into an array of numbers using ImageDataGenerator. Because I am using KFold cross validation, I will need to prepare our train data into an X and Y array.

In [30]:

```
# Creating labels for our train data and separating it into X and

def prepare_arrays(folder_name):
    train_files=os.listdir(input_path+'/train/'+folder_name)
    for i in train_files:
        X.append(i)
        for i in range(len(class_labels)):
            if(folder_name==class_labels[i]):
                Y.append(i)
```

In [31]:

```
# Assigning labels to images

for i in range(len(class_labels)):
    prepare_arrays(class_labels[i])
```

In [32]:

```
# Creating X and Y arrays on train data for cross validation

X=np.asarray(X)
Y=np.asarray(Y)
```

In [33]:

```
print(X.shape)
print(Y.shape)
```

```
(4591,)
(4591,)
```

In [34]:

```
# get all the data in the directory split/test (1146 images), and
test_generator = ImageDataGenerator(rescale=1./255).flow_from_dir
        'data/chest_xray/test',
        target_size=(64, 64), batch_size = 1146)

# get all the data in the directory split/validation (119 images)
val_generator = ImageDataGenerator(rescale=1./255).flow_from_dire
        'data/chest_xray/val',
        target_size=(64, 64), batch_size = 119)

# get all the data in the directory split/train (4591 images), an
train_generator = ImageDataGenerator(rescale=1./255).flow_from_di
        'data/chest_xray/train',
        target_size=(64, 64), batch_size=4591)
```

```
Found 1146 images belonging to 2 classes.
Found 119 images belonging to 2 classes.
```

```
Found 4591 images belonging to 2 classes.
```

In [35]:

```
# Separate our images into 2 sets of arrays:
# Images-translates image to numeric format
# Labels- labels image as 0 or 1 (normal vs pneumonia)

train_images, train_labels = next(train_generator)
test_images, test_labels = next(test_generator)
val_images, val_labels = next(val_generator)
```

In [36]:

```
# To illustrate information about each array created

m_train = train_images.shape[0]
num_px = train_images.shape[1]
m_test = test_images.shape[0]
m_val = val_images.shape[0]

print ("Number of training samples: " + str(m_train))
print ("Number of testing samples: " + str(m_test))
print ("Number of validation samples: " + str(m_val))
print ("train_images shape: " + str(train_images.shape))
print ("train_labels shape: " + str(train_labels.shape))
print ("test_images shape: " + str(test_images.shape))
print ("test_labels shape: " + str(test_labels.shape))
print ("val_images shape: " + str(val_images.shape))
print ("val_labels shape: " + str(val_labels.shape))
```

```
Number of training samples: 4591
Number of testing samples: 1146
Number of validation samples: 119
train_images shape: (4591, 64, 64, 3)
train_labels shape: (4591, 2)
test_images shape: (1146, 64, 64, 3)
test_labels shape: (1146, 2)
val_images shape: (119, 64, 64, 3)
val_labels shape: (119, 2)
```

In [37]:

```
# Collapsing image arrays into 2D array

X_train = train_images.reshape(train_images.shape[0], -1)
X_test = test_images.reshape(test_images.shape[0], -1)
X_val = val_images.reshape(val_images.shape[0], -1)

print(X_train.shape)
print(X_test.shape)
print(X_val.shape)
```

```
(4591, 12288)
(1146, 12288)
(119, 12288)
```

In [38]:

```
# Collapsing label arrays into 1D array

y_train = np.reshape(train_labels[:,0], (4591,1))
y_test = np.reshape(test_labels[:,0], (1146,1))
y_val = np.reshape(val_labels[:,0], (119,1))

print(y_train.shape)
print(y_test.shape)
print(y_val.shape)
```

```
(4591, 1)
(1146, 1)
(119, 1)
```

Section 4: Modeling

Now that our images have been pre-processed and transformed into arrays that we can work with, we can now run a baseline model. The main metric I will be using is accuracy, however, upon evaluation I will be focusing on recall as false negatives are the reason why pneumonia patients are readmitted to the hospital.

Baseline Model:

In [39]:

```
# Creating function to depict fit results and graph train accurac

def model_results(model, model_fit):

    model_func = model
    model_fit_func = model_fit
    results_train = model_func.evaluate(X_train, y_train)
    results_test = model_func.evaluate(X_test, y_test)

    model_val_dict = model_fit_func.history
    model_val_dict.keys()

    loss_values = model_val_dict['loss']
    val_loss_values = model_val_dict['val_loss']
    acc_values = model_val_dict['acc']
    val_acc_values = model_val_dict['val_acc']
    epochs = range(1, len(loss_values) + 1)

    fig, (ax, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(12,

    ax.plot(epochs, loss_values, label='Training loss')
    ax.plot(epochs, val_loss_values, label='Test loss')
    ax.set_title('Training & Testing loss')
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Loss')
```

```
    ax.set_ylabel('Loss')
    ax.legend();

    ax2.plot(epochs, acc_values, label='Training acc')
    ax2.plot(epochs, val_acc_values, label='Test acc')
    ax2.set_title('Training & Testing accuracy')
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Accuracy')
    ax2.legend();


    return print("\nTrain Results (Loss, Acc.):", results_train),
```

In [40]:

```
# Create simple baseline model to check accuracy and loss

baseline_model = models.Sequential()
baseline_model.add(layers.Dense(10, activation='relu', input_shap
baseline_model.add(layers.Dense(4, activation='sigmoid'))
baseline_model.add(layers.Dense(1, activation='sigmoid'))
baseline_model.compile(optimizer="sgd",
                       loss='binary_crossentropy',
                       metrics=['accuracy'])

baseline_model_fit = baseline_model.fit(X_train,
                                        y_train,
                                        epochs=50,
                                        batch_size=32,
                                        validation_data=(X_te
```

```
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 0s 97us/step - loss:
0.5418 - acc: 0.7299 - val_loss: 0.4956 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 0s 104us/step - loss
: 0.4574 - acc: 0.7737 - val_loss: 0.4246 - val_acc: 0.7967
Epoch 3/50
4591/4591 [==============================] - 0s 69us/step - loss:
0.3977 - acc: 0.8582 - val_loss: 0.3842 - val_acc: 0.9031
Epoch 4/50
4591/4591 [==============================] - 0s 64us/step - loss:
0.3568 - acc: 0.8874 - val_loss: 0.3493 - val_acc: 0.8674
Epoch 5/50
4591/4591 [==============================] - 0s 66us/step - loss:
0.3221 - acc: 0.9044 - val_loss: 0.3043 - val_acc: 0.9084
Epoch 6/50
4591/4591 [==============================] - 0s 60us/step - loss:
0.2960 - acc: 0.9103 - val_loss: 0.2756 - val_acc: 0.9206
Epoch 7/50
4591/4591 [==============================] - 0s 69us/step - loss:
0.2727 - acc: 0.9161 - val_loss: 0.2738 - val_acc: 0.9058
Epoch 8/50
4591/4591 [==============================] - 0s 88us/step - loss:
```

```
4591/4591 [==============================] - 0s [illegible]us/step - loss:
0.2620 - acc: 0.9170 - val_loss: 0.2490 - val_acc: 0.9293
Epoch 9/50
4591/4591 [==============================] - 0s 103us/step - loss
: 0.2552 - acc: 0.9174 - val_loss: 0.2777 - val_acc: 0.8997
Epoch 10/50
4591/4591 [==============================] - 0s 81us/step - loss:
0.2411 - acc: 0.9220 - val_loss: 0.2286 - val_acc: 0.9389
Epoch 11/50
4591/4591 [==============================] - 0s 76us/step - loss:
0.2376 - acc: 0.9233 - val_loss: 0.2231 - val_acc: 0.9407
Epoch 12/50
4591/4591 [==============================] - 0s 94us/step - loss:
0.2330 - acc: 0.9240 - val_loss: 0.2178 - val_acc: 0.9380
Epoch 13/50
4591/4591 [==============================] - 0s 92us/step - loss:
0.2242 - acc: 0.9279 - val_loss: 0.2165 - val_acc: 0.9337
Epoch 14/50
4591/4591 [==============================] - 0s 87us/step - loss:
0.2144 - acc: 0.9331 - val_loss: 0.2073 - val_acc: 0.9415
Epoch 15/50
4591/4591 [==============================] - 0s 103us/step - loss
: 0.2148 - acc: 0.9272 - val_loss: 0.2609 - val_acc: 0.8918
Epoch 16/50
4591/4591 [==============================] - 0s 96us/step - loss:
0.2117 - acc: 0.9283 - val_loss: 0.2016 - val_acc: 0.9398
Epoch 17/50
4591/4591 [==============================] - 0s 87us/step - loss:
0.2104 - acc: 0.9288 - val_loss: 0.2020 - val_acc: 0.9433
Epoch 18/50
4591/4591 [==============================] - 0s 60us/step - loss:
0.2048 - acc: 0.9325 - val_loss: 0.4560 - val_acc: 0.8281
Epoch 19/50
4591/4591 [==============================] - 0s 59us/step - loss:
0.2068 - acc: 0.9301 - val_loss: 0.4998 - val_acc: 0.8115
Epoch 20/50
4591/4591 [==============================] - 0s 64us/step - loss:
0.2108 - acc: 0.9259 - val_loss: 0.1932 - val_acc: 0.9407
Epoch 21/50
4591/4591 [==============================] - 0s 75us/step - loss:
0.2000 - acc: 0.9323 - val_loss: 0.1944 - val_acc: 0.9328
Epoch 22/50
4591/4591 [==============================] - 0s 70us/step - loss:
0.1972 - acc: 0.9340 - val_loss: 0.1892 - val_acc: 0.9380
Epoch 23/50
4591/4591 [==============================] - 0s 90us/step - loss:
0.1934 - acc: 0.9347 - val_loss: 0.2499 - val_acc: 0.9092
Epoch 24/50
4591/4591 [==============================] - 0s 74us/step - loss:
0.1946 - acc: 0.9344 - val_loss: 0.1871 - val_acc: 0.9372
Epoch 25/50
4591/4591 [==============================] - 0s 87us/step - loss:
0.1917 - acc: 0.9377 - val_loss: 0.2008 - val_acc: 0.9284
Epoch 26/50
```

```
4591/4591 [==============================] - 0s 91us/step - loss:
0.1908 - acc: 0.9351 - val_loss: 0.2055 - val_acc: 0.9284
Epoch 27/50
4591/4591 [==============================] - 0s 73us/step - loss:
0.1907 - acc: 0.9342 - val_loss: 0.2060 - val_acc: 0.9328
Epoch 28/50
4591/4591 [==============================] - 0s 65us/step - loss:
0.1865 - acc: 0.9351 - val_loss: 0.1842 - val_acc: 0.9415
Epoch 29/50
4591/4591 [==============================] - 0s 75us/step - loss:
0.1851 - acc: 0.9349 - val_loss: 0.1803 - val_acc: 0.9407
Epoch 30/50
4591/4591 [==============================] - 0s 68us/step - loss:
0.1773 - acc: 0.9397 - val_loss: 0.1931 - val_acc: 0.9337
Epoch 31/50
4591/4591 [==============================] - 0s 71us/step - loss:
0.1786 - acc: 0.9408 - val_loss: 0.1797 - val_acc: 0.9407
Epoch 32/50
4591/4591 [==============================] - 0s 61us/step - loss:
0.1755 - acc: 0.9403 - val_loss: 0.2261 - val_acc: 0.9171
Epoch 33/50
4591/4591 [==============================] - 0s 74us/step - loss:
0.1761 - acc: 0.9408 - val_loss: 0.3065 - val_acc: 0.8578
Epoch 34/50
4591/4591 [==============================] - 0s 66us/step - loss:
0.1822 - acc: 0.9375 - val_loss: 0.2245 - val_acc: 0.9232
Epoch 35/50
4591/4591 [==============================] - 0s 84us/step - loss:
0.1741 - acc: 0.9388 - val_loss: 0.1775 - val_acc: 0.9372
Epoch 36/50
4591/4591 [==============================] - 0s 71us/step - loss:
0.1778 - acc: 0.9392 - val_loss: 0.1757 - val_acc: 0.9389
Epoch 37/50
4591/4591 [==============================] - 0s 75us/step - loss:
0.1739 - acc: 0.9381 - val_loss: 0.2481 - val_acc: 0.8909
Epoch 38/50
4591/4591 [==============================] - 0s 67us/step - loss:
0.1708 - acc: 0.9418 - val_loss: 0.1827 - val_acc: 0.9363
Epoch 39/50
4591/4591 [==============================] - 0s 80us/step - loss:
0.1756 - acc: 0.9373 - val_loss: 0.1730 - val_acc: 0.9389
Epoch 40/50
4591/4591 [==============================] - 0s 80us/step - loss:
0.1672 - acc: 0.9436 - val_loss: 0.1918 - val_acc: 0.9354
Epoch 41/50
4591/4591 [==============================] - 0s 74us/step - loss:
0.1687 - acc: 0.9423 - val_loss: 0.1803 - val_acc: 0.9389
Epoch 42/50
4591/4591 [==============================] - 0s 84us/step - loss:
0.1596 - acc: 0.9490 - val_loss: 0.1732 - val_acc: 0.9398
Epoch 43/50
4591/4591 [==============================] - 0s 90us/step - loss:
0.1634 - acc: 0.9436 - val_loss: 0.1734 - val_acc: 0.9372
Epoch 44/50
```

```
4591/4591 [==============================] - 0s 67us/step - loss: 0.1615 - acc: 0.9440 - val_loss: 0.2271 - val_acc: 0.9180
Epoch 45/50
4591/4591 [==============================] - 0s 71us/step - loss: 0.1540 - acc: 0.9503 - val_loss: 0.1729 - val_acc: 0.9398
Epoch 46/50
4591/4591 [==============================] - 0s 62us/step - loss: 0.1629 - acc: 0.9429 - val_loss: 0.1741 - val_acc: 0.9389
Epoch 47/50
4591/4591 [==============================] - 0s 71us/step - loss: 0.1604 - acc: 0.9431 - val_loss: 0.1761 - val_acc: 0.9363
Epoch 48/50
4591/4591 [==============================] - 0s 59us/step - loss: 0.1608 - acc: 0.9464 - val_loss: 0.1830 - val_acc: 0.9415
Epoch 49/50
4591/4591 [==============================] - 0s 66us/step - loss: 0.1516 - acc: 0.9477 - val_loss: 0.1921 - val_acc: 0.9363
Epoch 50/50
4591/4591 [==============================] - 0s 67us/step - loss: 0.1580 - acc: 0.9462 - val_loss: 0.2131 - val_acc: 0.9258
```

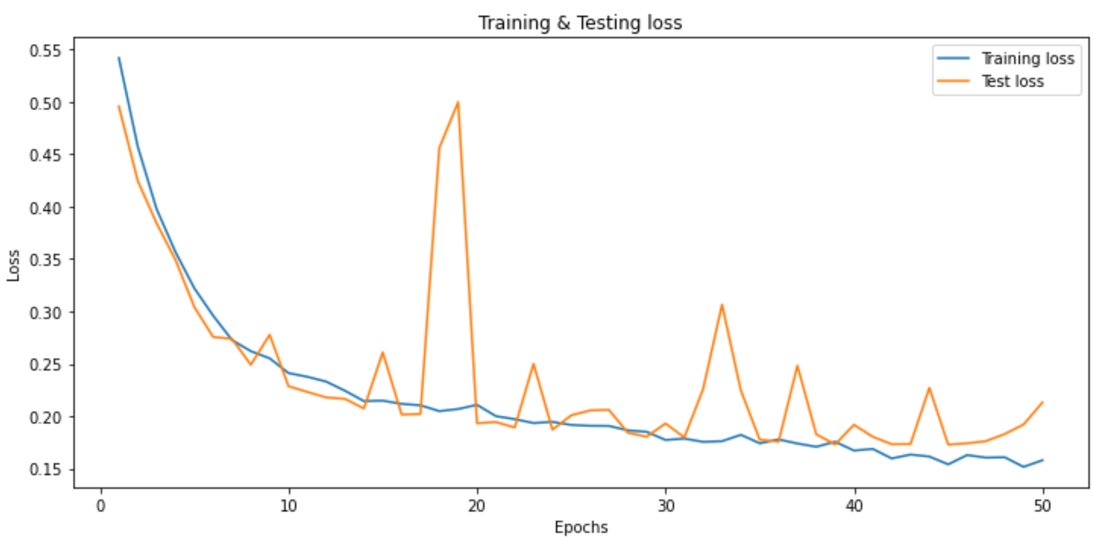
In [41]:

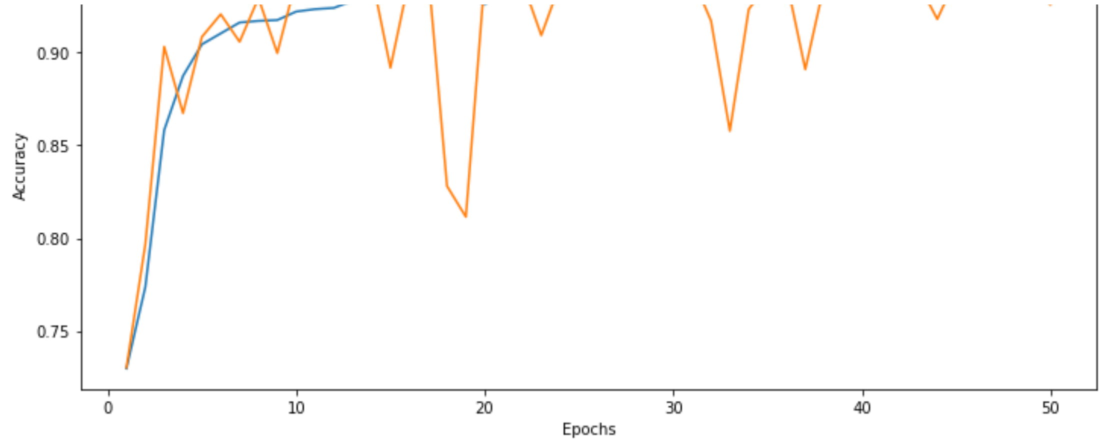
```python
# Call function for information

model_results(baseline_model,baseline_model_fit)
```

```
4591/4591 [==============================] - 0s 30us/step
1146/1146 [==============================] - 0s 30us/step

Train Results (Loss, Acc.): [0.17525029354620228, 0.9313874972902646]
Test Results (Loss, Acc.): [0.2131096540607291, 0.9258289703315882]
```

Out[41]:

```
(None, None)
```





As we can see above, our baseline model seems to be performing great in terms of accuracy and loss, and there doesn't appear to be overfitting between our train and test data. According to Francois Chollet, there are 4 ways we can reduce overfitting:

- Get more training data
- Reduce the capacity of the network
- Add weight regularization
- Add dropout

To see if our model performs better, I will add hidden layers, add regularizers, add dropout, and perform KFold Cross Validation.

Model 1 - Add Hidden Layers, Regularization, and Dropout:

In [42]:

```
# Creating new function comparing new model to baseline model

def compare_model_results(baseline_model, baseline_model_fit, moc

    model_baseline = baseline_model
    model_baseline_fit = baseline_model_fit
    model_func = model
    model_fit_func = model_fit
    results_train_baseline = model_baseline.evaluate(X_train, y_t
    results_test_baseline = model_baseline.evaluate(X_test, y_tes
    results_train_model = model_func.evaluate(X_train, y_train)
    results_test_model = model_func.evaluate(X_test, y_test)

    baseline_model_val_dict = model_baseline_fit.history
    baseline_model_val_dict.keys()
    model_val_dict = model_fit_func.history
    model_val_dict.keys()

    model_loss_values = model_val_dict['loss']
```

```
    model_val_loss_values = model_val_dict['val_loss']
    model_acc_values = model_val_dict['acc']
    model_val_acc_values = model_val_dict['val_acc']
    model_epochs = range(1, len(model_loss_values) + 1)

    baseline_loss_values = baseline_model_val_dict['loss']
    baseline_val_loss_values = baseline_model_val_dict['val_loss'
    baseline_acc_values = baseline_model_val_dict['acc']
    baseline_val_acc_values = baseline_model_val_dict['val_acc']
    baseline_epochs = range(1, len(baseline_loss_values) + 1)

    fig, (ax, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(20,

    ax.plot(model_epochs, model_loss_values, label='Training loss
    ax.plot(model_epochs, model_val_loss_values, label='Testing l
    ax.plot(baseline_epochs, baseline_loss_values, label='Trainin
    ax.plot(baseline_epochs, baseline_val_loss_values, label='Tes
    ax.set_title('Training & Testing Loss Current Model vs Baseli
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Loss')
    ax.legend();

    ax2.plot(model_epochs, model_acc_values, label='Training acc
    ax2.plot(model_epochs, model_val_acc_values, label='Testing a
    ax2.plot(baseline_epochs, baseline_acc_values, label='Trainin
    ax2.plot(baseline_epochs, baseline_val_acc_values, label='Tes
    ax2.set_title('Training & Testing Accuracy Current Model vs B
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Accuracy')
    ax2.legend();

    return print("\nBaseline Model Train Results (Loss, Acc.):",
```

In [43]:

```
# Using KFold Cross Validation to fit model into 5 folds. Checkin

kf = KFold(5, shuffle=True, random_state=123)
fold=0
fold_results_1 = []
for i in kf.split(X,Y):
    fold+=1
    print("Results for fold",fold)

    # More complex model with an extra hidden layer, Dropout, and
    model_1 = models.Sequential()
    model_1.add(layers.Dense(20, activation='relu', kernel_regula
    model_1.add(layers.Dense(20, activation='relu', kernel_regula
    model_1.add(layers.Dropout(0.5))
    model_1.add(layers.Dense(1, activation='sigmoid'))

    model_1.compile(optimizer="sgd",
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
```

```
    model_1_fit = model_1.fit(X_train,
                              y_train,
                              epochs=50,
                              batch_size=32,
                              validation_data=(X_test, y_test))
    compare_model_results(baseline_model,baseline_model_fit,model
    results_1 = {"Train Results:" : model_1.evaluate(X_train,y_tr
    fold_results_1.append(results_1)
```

```
Results for fold 1
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 1s 175us/step - los
: 0.6858 - acc: 0.6855 - val_loss: 0.5665 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.5312 - acc: 0.7195 - val_loss: 0.4273 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 0s 98us/step - loss
0.4684 - acc: 0.7271 - val_loss: 0.4304 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 0s 94us/step - loss
0.4402 - acc: 0.7271 - val_loss: 0.4418 - val_acc: 0.7304
Epoch 5/50
4591/4591 [==============================] - 1s 147us/step - los
: 0.4166 - acc: 0.8676 - val_loss: 0.3536 - val_acc: 0.9119
Epoch 6/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.3909 - acc: 0.8839 - val_loss: 0.3334 - val_acc: 0.9276
Epoch 7/50
4591/4591 [==============================] - 1s 138us/step - los
: 0.3605 - acc: 0.8769 - val_loss: 0.2796 - val_acc: 0.9354
Epoch 8/50
4591/4591 [==============================] - 1s 119us/step - los
: 0.3435 - acc: 0.8904 - val_loss: 0.2662 - val_acc: 0.9302
Epoch 9/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.3347 - acc: 0.8928 - val_loss: 0.3374 - val_acc: 0.8927
Epoch 10/50
4591/4591 [==============================] - 0s 91us/step - loss
0.3151 - acc: 0.9059 - val_loss: 0.2778 - val_acc: 0.9049
Epoch 11/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.3007 - acc: 0.9087 - val_loss: 0.2651 - val_acc: 0.9066
Epoch 12/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2989 - acc: 0.9129 - val_loss: 0.2576 - val_acc: 0.9319
Epoch 13/50
4591/4591 [==============================] - 0s 102us/step - los
: 0.2863 - acc: 0.9192 - val_loss: 0.3391 - val_acc: 0.8979
Epoch 14/50
4591/4591 [==============================] - 1s 109us/step - los
: 0.2865 - acc: 0.9124 - val_loss: 0.3310 - val_acc: 0.8944
```

```
Epoch 15/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.2817 - acc: 0.9183 - val_loss: 0.2547 - val_acc: 0.9154
Epoch 16/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2734 - acc: 0.9242 - val_loss: 0.2526 - val_acc: 0.9145
Epoch 17/50
4591/4591 [==============================] - 1s 135us/step - los
: 0.2675 - acc: 0.9255 - val_loss: 0.2804 - val_acc: 0.9180
Epoch 18/50
4591/4591 [==============================] - 1s 132us/step - los
: 0.2631 - acc: 0.9209 - val_loss: 0.2692 - val_acc: 0.9049
Epoch 19/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2695 - acc: 0.9188 - val_loss: 0.4520 - val_acc: 0.8508
Epoch 20/50
4591/4591 [==============================] - 0s 103us/step - los
: 0.2673 - acc: 0.9212 - val_loss: 0.2272 - val_acc: 0.9380
Epoch 21/50
4591/4591 [==============================] - 1s 136us/step - los
: 0.2638 - acc: 0.9259 - val_loss: 0.2747 - val_acc: 0.9250
Epoch 22/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2578 - acc: 0.9233 - val_loss: 0.2313 - val_acc: 0.9319
Epoch 23/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2555 - acc: 0.9303 - val_loss: 0.2321 - val_acc: 0.9328
Epoch 24/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2551 - acc: 0.9242 - val_loss: 0.2513 - val_acc: 0.9328
Epoch 25/50
4591/4591 [==============================] - 1s 126us/step - los
: 0.2452 - acc: 0.9266 - val_loss: 0.2267 - val_acc: 0.9398
Epoch 26/50
4591/4591 [==============================] - 1s 112us/step - los
: 0.2551 - acc: 0.9229 - val_loss: 0.2765 - val_acc: 0.9197
Epoch 27/50
4591/4591 [==============================] - 1s 121us/step - los
: 0.2490 - acc: 0.9275 - val_loss: 0.2256 - val_acc: 0.9372
Epoch 28/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.2557 - acc: 0.9294 - val_loss: 0.2551 - val_acc: 0.9241
Epoch 29/50
4591/4591 [==============================] - 1s 110us/step - los
: 0.2580 - acc: 0.9227 - val_loss: 0.2350 - val_acc: 0.9346
Epoch 30/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2397 - acc: 0.9303 - val_loss: 0.2778 - val_acc: 0.9197
Epoch 31/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2406 - acc: 0.9301 - val_loss: 0.2279 - val_acc: 0.9354
Epoch 32/50
4591/4591 [==============================] - 0s 103us/step - los
```

```
: 0.2333 - acc: 0.9344 - val_loss: 0.4111 - val_acc: 0.8464
Epoch 33/50
4591/4591 [==============================] - 0s 96us/step - loss
0.2493 - acc: 0.9296 - val_loss: 0.2259 - val_acc: 0.9380
Epoch 34/50
4591/4591 [==============================] - 1s 133us/step - los
: 0.2480 - acc: 0.9288 - val_loss: 0.2291 - val_acc: 0.9372
Epoch 35/50
4591/4591 [==============================] - 0s 93us/step - loss
0.2368 - acc: 0.9338 - val_loss: 0.2230 - val_acc: 0.9389
Epoch 36/50
4591/4591 [==============================] - 0s 96us/step - loss
0.2467 - acc: 0.9288 - val_loss: 0.2217 - val_acc: 0.9407
Epoch 37/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2382 - acc: 0.9355 - val_loss: 0.2238 - val_acc: 0.9389
Epoch 38/50
4591/4591 [==============================] - 0s 109us/step - los
: 0.2349 - acc: 0.9340 - val_loss: 0.2410 - val_acc: 0.9319
Epoch 39/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2262 - acc: 0.9355 - val_loss: 0.2226 - val_acc: 0.9389
Epoch 40/50
4591/4591 [==============================] - 0s 98us/step - loss
0.2347 - acc: 0.9296 - val_loss: 0.2368 - val_acc: 0.9372
Epoch 41/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2341 - acc: 0.9320 - val_loss: 0.2343 - val_acc: 0.9363
Epoch 42/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2260 - acc: 0.9329 - val_loss: 0.2382 - val_acc: 0.9311
Epoch 43/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2273 - acc: 0.9349 - val_loss: 0.2325 - val_acc: 0.9328
Epoch 44/50
4591/4591 [==============================] - 0s 108us/step - los
: 0.2385 - acc: 0.9314 - val_loss: 0.2220 - val_acc: 0.9380
Epoch 45/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2196 - acc: 0.9408 - val_loss: 0.3075 - val_acc: 0.8813
Epoch 46/50
4591/4591 [==============================] - 0s 91us/step - loss
0.2339 - acc: 0.9360 - val_loss: 0.2873 - val_acc: 0.9023
Epoch 47/50
4591/4591 [==============================] - 0s 102us/step - los
: 0.2267 - acc: 0.9399 - val_loss: 0.2348 - val_acc: 0.9354
Epoch 48/50
4591/4591 [==============================] - 0s 100us/step - los
: 0.2217 - acc: 0.9403 - val_loss: 0.2330 - val_acc: 0.9363
Epoch 49/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2332 - acc: 0.9355 - val_loss: 0.2617 - val_acc: 0.9319
Epoch 50/50
4591/4591 [==============================] - 0s 106us/step - los
```

```
: 0.2224 - acc: 0.9408 - val_loss: 0.2293 - val_acc: 0.9363
4591/4591 [==============================] - 0s 32us/step
1146/1146 [==============================] - 0s 29us/step
4591/4591 [==============================] - 0s 42us/step
1146/1146 [==============================] - 0s 46us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.18508577658067574,
.9501197996079286]
Current Model Test Results (Loss, Acc.): [0.2292993969006064, 0.
363001738959374]
4591/4591 [==============================] - 0s 40us/step
1146/1146 [==============================] - 0s 41us/step
Results for fold 2
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 1s 132us/step - los
: 0.6162 - acc: 0.7129 - val_loss: 0.5609 - val_acc: 0.8368
Epoch 2/50
4591/4591 [==============================] - 0s 91us/step - loss
0.5042 - acc: 0.7471 - val_loss: 0.4255 - val_acc: 0.7644
Epoch 3/50
4591/4591 [==============================] - 0s 90us/step - loss
0.4463 - acc: 0.8403 - val_loss: 0.5591 - val_acc: 0.7461
Epoch 4/50
4591/4591 [==============================] - 1s 137us/step - los
: 0.4223 - acc: 0.8667 - val_loss: 0.3592 - val_acc: 0.8569
Epoch 5/50
4591/4591 [==============================] - 1s 144us/step - los
: 0.3895 - acc: 0.8924 - val_loss: 0.3306 - val_acc: 0.9049
Epoch 6/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.3730 - acc: 0.8870 - val_loss: 0.3974 - val_acc: 0.8656
Epoch 7/50
4591/4591 [==============================] - 1s 130us/step - los
: 0.3571 - acc: 0.8917 - val_loss: 0.4073 - val_acc: 0.8403
Epoch 8/50
4591/4591 [==============================] - 1s 129us/step - los
: 0.3478 - acc: 0.8959 - val_loss: 0.3344 - val_acc: 0.9066
Epoch 9/50
4591/4591 [==============================] - 1s 137us/step - los
: 0.3400 - acc: 0.8952 - val_loss: 0.3067 - val_acc: 0.9171
Epoch 10/50
4591/4591 [==============================] - 1s 109us/step - los
: 0.3563 - acc: 0.8832 - val_loss: 0.2802 - val_acc: 0.9241
Epoch 11/50
4591/4591 [==============================] - 1s 126us/step - los
: 0.3389 - acc: 0.8974 - val_loss: 0.2762 - val_acc: 0.9232
Epoch 12/50
```

```
4591/4591 [==============================] - 1s 130us/step - los
: 0.3221 - acc: 0.9000 - val_loss: 0.2648 - val_acc: 0.9380
Epoch 13/50
4591/4591 [==============================] - 1s 119us/step - los
: 0.3089 - acc: 0.9105 - val_loss: 0.2625 - val_acc: 0.9363
Epoch 14/50
4591/4591 [==============================] - 1s 118us/step - los
: 0.3090 - acc: 0.9046 - val_loss: 0.2575 - val_acc: 0.9372
Epoch 15/50
4591/4591 [==============================] - 1s 131us/step - los
: 0.3157 - acc: 0.9059 - val_loss: 0.3534 - val_acc: 0.9023
Epoch 16/50
4591/4591 [==============================] - 1s 138us/step - los
: 0.3026 - acc: 0.9087 - val_loss: 0.3353 - val_acc: 0.8822
Epoch 17/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.3107 - acc: 0.9029 - val_loss: 0.2691 - val_acc: 0.9145
Epoch 18/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.3000 - acc: 0.9096 - val_loss: 0.2610 - val_acc: 0.9293
Epoch 19/50
4591/4591 [==============================] - 1s 120us/step - los
: 0.2990 - acc: 0.9120 - val_loss: 0.2603 - val_acc: 0.9284
Epoch 20/50
4591/4591 [==============================] - 1s 122us/step - los
: 0.2821 - acc: 0.9198 - val_loss: 0.2443 - val_acc: 0.9415
Epoch 21/50
4591/4591 [==============================] - 0s 103us/step - los
: 0.2896 - acc: 0.9109 - val_loss: 0.2931 - val_acc: 0.9031
Epoch 22/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2885 - acc: 0.9238 - val_loss: 0.2449 - val_acc: 0.9346
Epoch 23/50
4591/4591 [==============================] - 1s 113us/step - los
: 0.2734 - acc: 0.9196 - val_loss: 0.3163 - val_acc: 0.8988
Epoch 24/50
4591/4591 [==============================] - 1s 129us/step - los
: 0.2806 - acc: 0.9166 - val_loss: 0.8374 - val_acc: 0.6134
Epoch 25/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2820 - acc: 0.9120 - val_loss: 0.3514 - val_acc: 0.8962
Epoch 26/50
4591/4591 [==============================] - 1s 134us/step - los
: 0.2747 - acc: 0.9157 - val_loss: 0.2409 - val_acc: 0.9311
Epoch 27/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.2798 - acc: 0.9140 - val_loss: 0.2338 - val_acc: 0.9398
Epoch 28/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2691 - acc: 0.9244 - val_loss: 0.4213 - val_acc: 0.8438
Epoch 29/50
4591/4591 [==============================] - 0s 100us/step - los
: 0.2593 - acc: 0.9255 - val_loss: 0.2801 - val_acc: 0.9188
Epoch 30/50
```

```
4591/4591 [==============================] - 1s 119us/step - los
: 0.2688 - acc: 0.9212 - val_loss: 0.2559 - val_acc: 0.9188
Epoch 31/50
4591/4591 [==============================] - 1s 109us/step - los
: 0.2623 - acc: 0.9242 - val_loss: 0.2400 - val_acc: 0.9354
Epoch 32/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2636 - acc: 0.9242 - val_loss: 0.2914 - val_acc: 0.9328
Epoch 33/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2636 - acc: 0.9188 - val_loss: 0.2426 - val_acc: 0.9319
Epoch 34/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2561 - acc: 0.9225 - val_loss: 0.2306 - val_acc: 0.9398
Epoch 35/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.2535 - acc: 0.9303 - val_loss: 0.2461 - val_acc: 0.9311
Epoch 36/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2528 - acc: 0.9272 - val_loss: 0.3084 - val_acc: 0.8953
Epoch 37/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.2598 - acc: 0.9209 - val_loss: 0.2494 - val_acc: 0.9267
Epoch 38/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2497 - acc: 0.9318 - val_loss: 0.2691 - val_acc: 0.9223
Epoch 39/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2580 - acc: 0.9262 - val_loss: 0.2822 - val_acc: 0.9319
Epoch 40/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2518 - acc: 0.9296 - val_loss: 0.3191 - val_acc: 0.8831
Epoch 41/50
4591/4591 [==============================] - 1s 113us/step - los
: 0.2442 - acc: 0.9296 - val_loss: 0.2910 - val_acc: 0.9014
Epoch 42/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2463 - acc: 0.9262 - val_loss: 0.2748 - val_acc: 0.9005
Epoch 43/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2481 - acc: 0.9314 - val_loss: 0.2374 - val_acc: 0.9407
Epoch 44/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.2465 - acc: 0.9318 - val_loss: 0.3162 - val_acc: 0.8988
Epoch 45/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2327 - acc: 0.9351 - val_loss: 0.3155 - val_acc: 0.8997
Epoch 46/50
4591/4591 [==============================] - 1s 126us/step - los
: 0.2494 - acc: 0.9296 - val_loss: 0.4837 - val_acc: 0.8272
Epoch 47/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2437 - acc: 0.9279 - val_loss: 0.2245 - val_acc: 0.9372
Epoch 48/50
```

```
Epoch 48/50
4591/4591 [==============================] - 0s 108us/step - los
: 0.2433 - acc: 0.9303 - val_loss: 0.2708 - val_acc: 0.9250
Epoch 49/50
4591/4591 [==============================] - 1s 122us/step - los
: 0.2356 - acc: 0.9357 - val_loss: 0.2310 - val_acc: 0.9250
Epoch 50/50
4591/4591 [==============================] - 1s 112us/step - los
: 0.2370 - acc: 0.9344 - val_loss: 0.2469 - val_acc: 0.9311
4591/4591 [==============================] - 0s 35us/step
1146/1146 [==============================] - 0s 34us/step
4591/4591 [==============================] - 0s 45us/step
1146/1146 [==============================] - 0s 42us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.19995980290929313,
.9488128947941625]
Current Model Test Results (Loss, Acc.): [0.24686214424032607, 0
9310645718016965]
4591/4591 [==============================] - 0s 44us/step
1146/1146 [==============================] - 0s 55us/step
Results for fold 3
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 1s 152us/step - los
: 0.6072 - acc: 0.7155 - val_loss: 0.4549 - val_acc: 0.7784
Epoch 2/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.4941 - acc: 0.8046 - val_loss: 0.3617 - val_acc: 0.9066
Epoch 3/50
4591/4591 [==============================] - 1s 158us/step - los
: 0.4064 - acc: 0.8565 - val_loss: 0.6750 - val_acc: 0.7670
Epoch 4/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.3839 - acc: 0.8645 - val_loss: 0.2782 - val_acc: 0.9232
Epoch 5/50
4591/4591 [==============================] - 0s 108us/step - los
: 0.3496 - acc: 0.8891 - val_loss: 0.2602 - val_acc: 0.9293
Epoch 6/50
4591/4591 [==============================] - 1s 131us/step - los
: 0.3309 - acc: 0.8911 - val_loss: 0.4069 - val_acc: 0.8517
Epoch 7/50
4591/4591 [==============================] - 1s 131us/step - los
: 0.3043 - acc: 0.9037 - val_loss: 0.2503 - val_acc: 0.9232
Epoch 8/50
4591/4591 [==============================] - 1s 109us/step - los
: 0.2823 - acc: 0.9168 - val_loss: 0.4148 - val_acc: 0.8473
Epoch 9/50
4591/4591 [==============================] - 0s 103us/step - los
: 0.2933 - acc: 0.9142 - val_loss: 0.2391 - val_acc: 0.9363
```

```
Epoch 10/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2836 - acc: 0.9164 - val_loss: 0.2373 - val_acc: 0.9267
Epoch 11/50
4591/4591 [==============================] - 1s 131us/step - los
: 0.2596 - acc: 0.9253 - val_loss: 0.2693 - val_acc: 0.9197
Epoch 12/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2661 - acc: 0.9220 - val_loss: 0.2378 - val_acc: 0.9415
Epoch 13/50
4591/4591 [==============================] - 1s 119us/step - los
: 0.2721 - acc: 0.9246 - val_loss: 0.2752 - val_acc: 0.9188
Epoch 14/50
4591/4591 [==============================] - 1s 144us/step - los
: 0.2668 - acc: 0.9218 - val_loss: 0.2326 - val_acc: 0.9442
Epoch 15/50
4591/4591 [==============================] - 0s 107us/step - los
: 0.2619 - acc: 0.9238 - val_loss: 0.2274 - val_acc: 0.9398
Epoch 16/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.2663 - acc: 0.9242 - val_loss: 0.2563 - val_acc: 0.9267
Epoch 17/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2561 - acc: 0.9246 - val_loss: 0.2328 - val_acc: 0.9302
Epoch 18/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.2631 - acc: 0.9286 - val_loss: 0.2360 - val_acc: 0.9319
Epoch 19/50
4591/4591 [==============================] - 1s 118us/step - los
: 0.2613 - acc: 0.9255 - val_loss: 0.2255 - val_acc: 0.9389
Epoch 20/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2490 - acc: 0.9338 - val_loss: 0.2461 - val_acc: 0.9328
Epoch 21/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2497 - acc: 0.9307 - val_loss: 0.2706 - val_acc: 0.9232
Epoch 22/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.2565 - acc: 0.9266 - val_loss: 0.3098 - val_acc: 0.9014
Epoch 23/50
4591/4591 [==============================] - 1s 134us/step - los
: 0.2618 - acc: 0.9275 - val_loss: 0.2241 - val_acc: 0.9407
Epoch 24/50
4591/4591 [==============================] - 1s 118us/step - los
: 0.2539 - acc: 0.9277 - val_loss: 0.2473 - val_acc: 0.9319
Epoch 25/50
4591/4591 [==============================] - 1s 111us/step - los
: 0.2424 - acc: 0.9329 - val_loss: 0.2676 - val_acc: 0.9119
Epoch 26/50
4591/4591 [==============================] - 1s 124us/step - los
: 0.2312 - acc: 0.9384 - val_loss: 0.2523 - val_acc: 0.9302
Epoch 27/50
4591/4591 [==============================] - 1s 133us/step - los
: 0.2535 - acc: 0.9272 - val_loss: 0.2547 - val_acc: 0.9311
```

```
: 0.2535 - acc: 0.9272 - val_loss: 0.2347 - val_acc: 0.9311
Epoch 28/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2429 - acc: 0.9344 - val_loss: 0.2272 - val_acc: 0.9407
Epoch 29/50
4591/4591 [==============================] - 1s 110us/step - los
: 0.2429 - acc: 0.9386 - val_loss: 0.3181 - val_acc: 0.9014
Epoch 30/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2510 - acc: 0.9296 - val_loss: 0.2830 - val_acc: 0.8997
Epoch 31/50
4591/4591 [==============================] - 1s 122us/step - los
: 0.2357 - acc: 0.9318 - val_loss: 0.2438 - val_acc: 0.9363
Epoch 32/50
4591/4591 [==============================] - 1s 126us/step - los
: 0.2397 - acc: 0.9340 - val_loss: 0.3243 - val_acc: 0.8997
Epoch 33/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2276 - acc: 0.9362 - val_loss: 0.2211 - val_acc: 0.9424
Epoch 34/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2393 - acc: 0.9349 - val_loss: 0.2193 - val_acc: 0.9407
Epoch 35/50
4591/4591 [==============================] - 1s 138us/step - los
: 0.2257 - acc: 0.9368 - val_loss: 0.2216 - val_acc: 0.9354
Epoch 36/50
4591/4591 [==============================] - 1s 129us/step - los
: 0.2377 - acc: 0.9373 - val_loss: 0.2803 - val_acc: 0.9084
Epoch 37/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2376 - acc: 0.9353 - val_loss: 0.2171 - val_acc: 0.9407
Epoch 38/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2215 - acc: 0.9418 - val_loss: 0.2180 - val_acc: 0.9407
Epoch 39/50
4591/4591 [==============================] - 0s 98us/step - loss
0.2308 - acc: 0.9364 - val_loss: 0.2354 - val_acc: 0.9354
Epoch 40/50
4591/4591 [==============================] - 1s 132us/step - los
: 0.2298 - acc: 0.9371 - val_loss: 0.2493 - val_acc: 0.9337
Epoch 41/50
4591/4591 [==============================] - 1s 136us/step - los
: 0.2271 - acc: 0.9362 - val_loss: 0.4582 - val_acc: 0.8586
Epoch 42/50
4591/4591 [==============================] - 1s 119us/step - los
: 0.2249 - acc: 0.9392 - val_loss: 0.2227 - val_acc: 0.9372
Epoch 43/50
4591/4591 [==============================] - 1s 112us/step - los
: 0.2321 - acc: 0.9394 - val_loss: 0.2170 - val_acc: 0.9389
Epoch 44/50
4591/4591 [==============================] - 1s 111us/step - los
: 0.2216 - acc: 0.9408 - val_loss: 0.2251 - val_acc: 0.9363
Epoch 45/50
4591/4591 [==============================] - 0s 96us/step - loss
```

```
0.2336 - acc: 0.9362 - val_loss: 0.2759 - val_acc: 0.9145
Epoch 46/50
4591/4591 [==============================] - 1s 121us/step - los
: 0.2282 - acc: 0.9399 - val_loss: 0.2223 - val_acc: 0.9372
Epoch 47/50
4591/4591 [==============================] - 1s 111us/step - los
: 0.2192 - acc: 0.9412 - val_loss: 0.2420 - val_acc: 0.9354
Epoch 48/50
4591/4591 [==============================] - 0s 98us/step - loss
0.2141 - acc: 0.9436 - val_loss: 0.2982 - val_acc: 0.9127
Epoch 49/50
4591/4591 [==============================] - 1s 133us/step - los
: 0.2107 - acc: 0.9458 - val_loss: 0.2191 - val_acc: 0.9415
Epoch 50/50
4591/4591 [==============================] - 1s 127us/step - los
: 0.2125 - acc: 0.9394 - val_loss: 0.2718 - val_acc: 0.9119
4591/4591 [==============================] - 0s 34us/step
1146/1146 [==============================] - 0s 38us/step
4591/4591 [==============================] - 0s 46us/step
1146/1146 [==============================] - 0s 45us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.24837131513616562,
.9194075365233748]
Current Model Test Results (Loss, Acc.): [0.27179394040848365, 0
9118673641228134]
4591/4591 [==============================] - 0s 46us/step
1146/1146 [==============================] - 0s 47us/step
Results for fold 4
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 1s 166us/step - los
: 0.6621 - acc: 0.6611 - val_loss: 0.5501 - val_acc: 0.8124
Epoch 2/50
4591/4591 [==============================] - 0s 107us/step - los
: 0.5277 - acc: 0.7436 - val_loss: 0.7521 - val_acc: 0.5384
Epoch 3/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.4487 - acc: 0.8227 - val_loss: 0.3142 - val_acc: 0.9040
Epoch 4/50
4591/4591 [==============================] - 0s 99us/step - loss
0.4018 - acc: 0.8549 - val_loss: 0.3577 - val_acc: 0.8778
Epoch 5/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.3793 - acc: 0.8793 - val_loss: 0.3782 - val_acc: 0.8682
Epoch 6/50
4591/4591 [==============================] - 1s 138us/step - los
: 0.3447 - acc: 0.8992 - val_loss: 0.2719 - val_acc: 0.9154
Epoch 7/50
4591/4591 [==============================] - 1s 119us/step - los
```

```
: 0.3451 - acc: 0.8917 - val_loss: 0.3258 - val_acc: 0.8805
Epoch 8/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.3371 - acc: 0.8981 - val_loss: 0.2848 - val_acc: 0.9075
Epoch 9/50
4591/4591 [==============================] - 0s 95us/step - loss
0.3130 - acc: 0.9044 - val_loss: 0.4667 - val_acc: 0.8316
Epoch 10/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.3014 - acc: 0.9137 - val_loss: 0.2330 - val_acc: 0.9380
Epoch 11/50
4591/4591 [==============================] - 0s 109us/step - los
: 0.2918 - acc: 0.9151 - val_loss: 0.4473 - val_acc: 0.8211
Epoch 12/50
4591/4591 [==============================] - 0s 107us/step - los
: 0.2798 - acc: 0.9207 - val_loss: 0.2258 - val_acc: 0.9398
Epoch 13/50
4591/4591 [==============================] - 1s 121us/step - los
: 0.2861 - acc: 0.9164 - val_loss: 0.2382 - val_acc: 0.9232
Epoch 14/50
4591/4591 [==============================] - 1s 109us/step - los
: 0.2815 - acc: 0.9192 - val_loss: 0.2292 - val_acc: 0.9424
Epoch 15/50
4591/4591 [==============================] - 1s 110us/step - los
: 0.2728 - acc: 0.9275 - val_loss: 0.2560 - val_acc: 0.9127
Epoch 16/50
4591/4591 [==============================] - 1s 146us/step - los
: 0.2710 - acc: 0.9198 - val_loss: 0.2270 - val_acc: 0.9450
Epoch 17/50
4591/4591 [==============================] - 1s 113us/step - los
: 0.2727 - acc: 0.9292 - val_loss: 0.2287 - val_acc: 0.9442
Epoch 18/50
4591/4591 [==============================] - 0s 103us/step - los
: 0.2728 - acc: 0.9174 - val_loss: 0.5720 - val_acc: 0.7740
Epoch 19/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2716 - acc: 0.9242 - val_loss: 0.3733 - val_acc: 0.8639
Epoch 20/50
4591/4591 [==============================] - 0s 100us/step - los
: 0.2671 - acc: 0.9264 - val_loss: 0.3229 - val_acc: 0.8909
Epoch 21/50
4591/4591 [==============================] - 1s 119us/step - los
: 0.2694 - acc: 0.9272 - val_loss: 0.2316 - val_acc: 0.9328
Epoch 22/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2604 - acc: 0.9275 - val_loss: 0.2285 - val_acc: 0.9346
Epoch 23/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2619 - acc: 0.9290 - val_loss: 0.2308 - val_acc: 0.9328
Epoch 24/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2536 - acc: 0.9266 - val_loss: 0.2275 - val_acc: 0.9389
Epoch 25/50
```

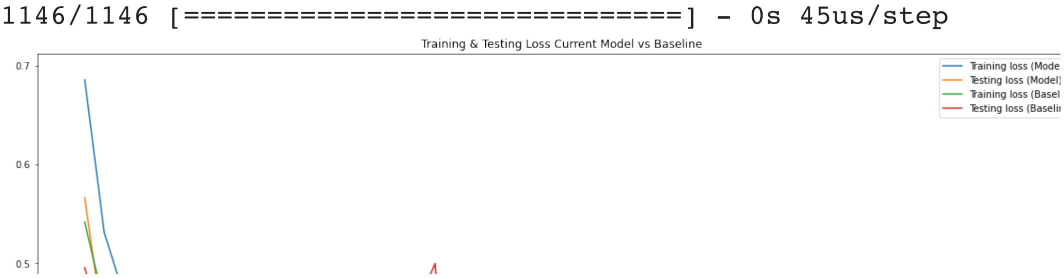
```
4591/4591 [==============================] - 0s 97us/step - loss
0.2600 - acc: 0.9283 - val_loss: 0.2573 - val_acc: 0.9276
Epoch 26/50
4591/4591 [==============================] - 0s 108us/step - los
: 0.2539 - acc: 0.9333 - val_loss: 0.2256 - val_acc: 0.9380
Epoch 27/50
4591/4591 [==============================] - 1s 143us/step - los
: 0.2530 - acc: 0.9264 - val_loss: 0.3724 - val_acc: 0.8560
Epoch 28/50
4591/4591 [==============================] - 1s 111us/step - los
: 0.2560 - acc: 0.9275 - val_loss: 0.2215 - val_acc: 0.9389
Epoch 29/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2583 - acc: 0.9272 - val_loss: 0.2227 - val_acc: 0.9398
Epoch 30/50
4591/4591 [==============================] - 1s 123us/step - los
: 0.2480 - acc: 0.9320 - val_loss: 0.2190 - val_acc: 0.9407
Epoch 31/50
4591/4591 [==============================] - 1s 124us/step - los
: 0.2441 - acc: 0.9323 - val_loss: 0.2649 - val_acc: 0.9215
Epoch 32/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.2441 - acc: 0.9325 - val_loss: 0.2407 - val_acc: 0.9346
Epoch 33/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2415 - acc: 0.9340 - val_loss: 0.2166 - val_acc: 0.9415
Epoch 34/50
4591/4591 [==============================] - 1s 121us/step - los
: 0.2323 - acc: 0.9366 - val_loss: 0.2203 - val_acc: 0.9346
Epoch 35/50
4591/4591 [==============================] - 1s 131us/step - los
: 0.2390 - acc: 0.9340 - val_loss: 0.2290 - val_acc: 0.9337
Epoch 36/50
4591/4591 [==============================] - 0s 98us/step - loss
0.2356 - acc: 0.9362 - val_loss: 0.2159 - val_acc: 0.9424
Epoch 37/50
4591/4591 [==============================] - 0s 102us/step - los
: 0.2346 - acc: 0.9401 - val_loss: 0.2140 - val_acc: 0.9424
Epoch 38/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2348 - acc: 0.9401 - val_loss: 0.2422 - val_acc: 0.9346
Epoch 39/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2375 - acc: 0.9347 - val_loss: 0.2141 - val_acc: 0.9415
Epoch 40/50
4591/4591 [==============================] - 1s 116us/step - los
: 0.2309 - acc: 0.9347 - val_loss: 0.2601 - val_acc: 0.9302
Epoch 41/50
4591/4591 [==============================] - 1s 115us/step - los
: 0.2375 - acc: 0.9357 - val_loss: 0.2749 - val_acc: 0.9188
Epoch 42/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2285 - acc: 0.9421 - val_loss: 0.2140 - val_acc: 0.9398
Epoch 43/50
```

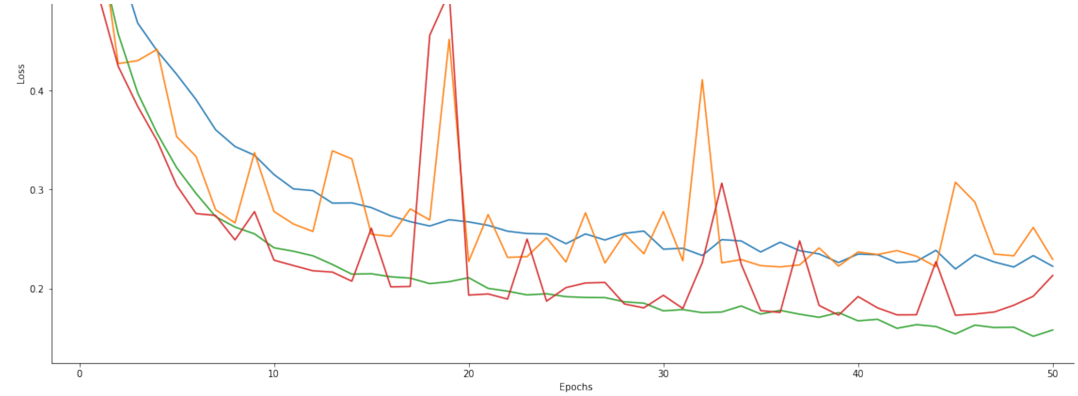
```
4591/4591 [==============================] - 1s 110us/step - los
: 0.2373 - acc: 0.9329 - val_loss: 0.2588 - val_acc: 0.9084
Epoch 44/50
4591/4591 [==============================] - 1s 132us/step - los
: 0.2342 - acc: 0.9362 - val_loss: 0.3643 - val_acc: 0.8735
Epoch 45/50
4591/4591 [==============================] - 1s 120us/step - los
: 0.2286 - acc: 0.9440 - val_loss: 0.2255 - val_acc: 0.9372
Epoch 46/50
4591/4591 [==============================] - 1s 121us/step - los
: 0.2245 - acc: 0.9357 - val_loss: 0.2232 - val_acc: 0.9380
Epoch 47/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.2152 - acc: 0.9408 - val_loss: 0.3086 - val_acc: 0.9040
Epoch 48/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2237 - acc: 0.9390 - val_loss: 0.2942 - val_acc: 0.9154
Epoch 49/50
4591/4591 [==============================] - 1s 125us/step - los
: 0.2204 - acc: 0.9353 - val_loss: 0.2188 - val_acc: 0.9389
Epoch 50/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2231 - acc: 0.9408 - val_loss: 0.2355 - val_acc: 0.9363
4591/4591 [==============================] - 0s 32us/step
1146/1146 [==============================] - 0s 30us/step
4591/4591 [==============================] - 0s 44us/step
1146/1146 [==============================] - 0s 50us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.20085735381805048,
.9431496406011762]
Current Model Test Results (Loss, Acc.): [0.23549430917486885, 0
9363001738959374]
4591/4591 [==============================] - 0s 46us/step
1146/1146 [==============================] - 0s 45us/step
Results for fold 5
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 1s 159us/step - los
: 0.6320 - acc: 0.6944 - val_loss: 0.4589 - val_acc: 0.7792
Epoch 2/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.4810 - acc: 0.8138 - val_loss: 0.3322 - val_acc: 0.8979
Epoch 3/50
4591/4591 [==============================] - 0s 100us/step - los
: 0.4117 - acc: 0.8606 - val_loss: 0.3088 - val_acc: 0.9162
Epoch 4/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.3674 - acc: 0.8837 - val_loss: 0.3218 - val_acc: 0.8892
```

```
Epoch 5/50
4591/4591 [==============================] - 1s 118us/step - los
: 0.3377 - acc: 0.9000 - val_loss: 0.3296 - val_acc: 0.8901
Epoch 6/50
4591/4591 [==============================] - 0s 109us/step - los
: 0.3299 - acc: 0.8989 - val_loss: 0.2580 - val_acc: 0.9328
Epoch 7/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.3076 - acc: 0.9109 - val_loss: 0.2412 - val_acc: 0.9372
Epoch 8/50
4591/4591 [==============================] - 0s 107us/step - los
: 0.3140 - acc: 0.9076 - val_loss: 0.2785 - val_acc: 0.9119
Epoch 9/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2779 - acc: 0.9198 - val_loss: 0.2981 - val_acc: 0.9049
Epoch 10/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2977 - acc: 0.9185 - val_loss: 0.2296 - val_acc: 0.9398
Epoch 11/50
4591/4591 [==============================] - 0s 96us/step - loss
0.2819 - acc: 0.9205 - val_loss: 0.2436 - val_acc: 0.9188
Epoch 12/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2774 - acc: 0.9218 - val_loss: 0.4965 - val_acc: 0.8072
Epoch 13/50
4591/4591 [==============================] - 1s 126us/step - los
: 0.2661 - acc: 0.9281 - val_loss: 0.2509 - val_acc: 0.9232
Epoch 14/50
4591/4591 [==============================] - 1s 142us/step - los
: 0.2733 - acc: 0.9209 - val_loss: 0.2892 - val_acc: 0.9075
Epoch 15/50
4591/4591 [==============================] - 1s 128us/step - los
: 0.2653 - acc: 0.9270 - val_loss: 0.2393 - val_acc: 0.9215
Epoch 16/50
4591/4591 [==============================] - 0s 107us/step - los
: 0.2666 - acc: 0.9235 - val_loss: 0.3564 - val_acc: 0.8656
Epoch 17/50
4591/4591 [==============================] - 0s 100us/step - los
: 0.2547 - acc: 0.9281 - val_loss: 0.2344 - val_acc: 0.9346
Epoch 18/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2544 - acc: 0.9257 - val_loss: 0.2312 - val_acc: 0.9302
Epoch 19/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2510 - acc: 0.9275 - val_loss: 0.3014 - val_acc: 0.9058
Epoch 20/50
4591/4591 [==============================] - 0s 98us/step - loss
0.2529 - acc: 0.9349 - val_loss: 0.2216 - val_acc: 0.9363
Epoch 21/50
4591/4591 [==============================] - 0s 92us/step - loss
0.2464 - acc: 0.9338 - val_loss: 0.2196 - val_acc: 0.9415
Epoch 22/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2503 - acc: 0.9288 - val_loss: 0.2698 - val_acc: 0.9284
```

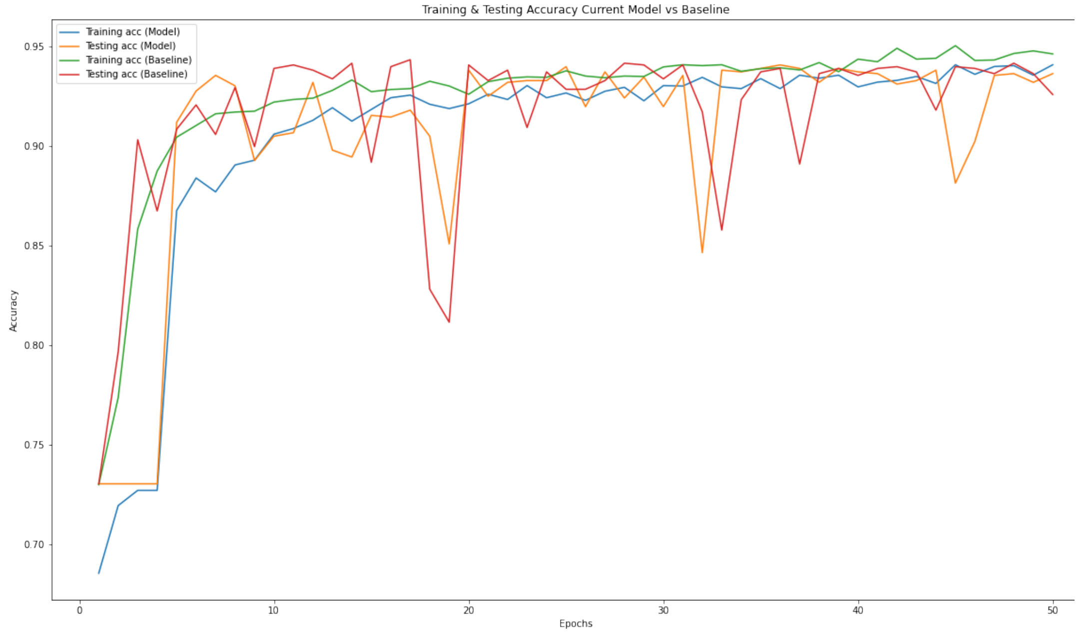
```
Epoch 23/50
4591/4591 [==============================] - 0s 103us/step - los
: 0.2611 - acc: 0.9251 - val_loss: 0.2397 - val_acc: 0.9328
Epoch 24/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2417 - acc: 0.9344 - val_loss: 0.2380 - val_acc: 0.9284
Epoch 25/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2382 - acc: 0.9386 - val_loss: 0.2224 - val_acc: 0.9398
Epoch 26/50
4591/4591 [==============================] - 1s 118us/step - los
: 0.2381 - acc: 0.9320 - val_loss: 0.2248 - val_acc: 0.9398
Epoch 27/50
4591/4591 [==============================] - 1s 132us/step - los
: 0.2334 - acc: 0.9366 - val_loss: 0.2208 - val_acc: 0.9433
Epoch 28/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2378 - acc: 0.9351 - val_loss: 0.2191 - val_acc: 0.9433
Epoch 29/50
4591/4591 [==============================] - 1s 114us/step - los
: 0.2317 - acc: 0.9355 - val_loss: 0.2861 - val_acc: 0.9154
Epoch 30/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2358 - acc: 0.9388 - val_loss: 0.2309 - val_acc: 0.9354
Epoch 31/50
4591/4591 [==============================] - 0s 91us/step - loss
0.2288 - acc: 0.9403 - val_loss: 0.2944 - val_acc: 0.8927
Epoch 32/50
4591/4591 [==============================] - 0s 101us/step - los
: 0.2338 - acc: 0.9368 - val_loss: 0.2587 - val_acc: 0.9319
Epoch 33/50
4591/4591 [==============================] - 0s 97us/step - loss
0.2396 - acc: 0.9347 - val_loss: 0.2194 - val_acc: 0.9354
Epoch 34/50
4591/4591 [==============================] - 0s 99us/step - loss
0.2386 - acc: 0.9299 - val_loss: 0.2441 - val_acc: 0.9319
Epoch 35/50
4591/4591 [==============================] - 0s 106us/step - los
: 0.2313 - acc: 0.9362 - val_loss: 0.2211 - val_acc: 0.9398
Epoch 36/50
4591/4591 [==============================] - 0s 104us/step - los
: 0.2273 - acc: 0.9425 - val_loss: 0.2294 - val_acc: 0.9328
Epoch 37/50
4591/4591 [==============================] - 0s 102us/step - los
: 0.2323 - acc: 0.9347 - val_loss: 0.3011 - val_acc: 0.8901
Epoch 38/50
4591/4591 [==============================] - 1s 120us/step - los
: 0.2198 - acc: 0.9405 - val_loss: 0.2146 - val_acc: 0.9398
Epoch 39/50
4591/4591 [==============================] - 0s 108us/step - los
: 0.2236 - acc: 0.9399 - val_loss: 0.2757 - val_acc: 0.9180
Epoch 40/50
4591/4591 [==============================] - 0s 105us/step - los
: 0.2217 - acc: 0.9379 - val_loss: 0.2565 - val_acc: 0.9293
```

```
: 0.2217 - acc: 0.9379 - val_loss: 0.2565 - val_acc: 0.9293
Epoch 41/50
4591/4591 [==============================] - 1s 112us/step - los
: 0.2228 - acc: 0.9397 - val_loss: 0.2223 - val_acc: 0.9363
Epoch 42/50
4591/4591 [==============================] - 1s 159us/step - los
: 0.2279 - acc: 0.9375 - val_loss: 0.2168 - val_acc: 0.9424
Epoch 43/50
4591/4591 [==============================] - 1s 110us/step - los
: 0.2139 - acc: 0.9418 - val_loss: 0.2540 - val_acc: 0.9337
Epoch 44/50
4591/4591 [==============================] - 1s 148us/step - los
: 0.2168 - acc: 0.9410 - val_loss: 0.3269 - val_acc: 0.9023
Epoch 45/50
4591/4591 [==============================] - 1s 130us/step - los
: 0.2175 - acc: 0.9371 - val_loss: 0.2172 - val_acc: 0.9354
Epoch 46/50
4591/4591 [==============================] - 1s 134us/step - los
: 0.2245 - acc: 0.9381 - val_loss: 0.2158 - val_acc: 0.9363
Epoch 47/50
4591/4591 [==============================] - 1s 112us/step - los
: 0.2268 - acc: 0.9403 - val_loss: 0.2563 - val_acc: 0.9319
Epoch 48/50
4591/4591 [==============================] - 1s 133us/step - los
: 0.2179 - acc: 0.9429 - val_loss: 0.2545 - val_acc: 0.9223
Epoch 49/50
4591/4591 [==============================] - 1s 117us/step - los
: 0.2095 - acc: 0.9449 - val_loss: 0.2712 - val_acc: 0.9258
Epoch 50/50
4591/4591 [==============================] - 1s 124us/step - los
: 0.2200 - acc: 0.9401 - val_loss: 0.2789 - val_acc: 0.9040
4591/4591 [==============================] - 0s 52us/step
1146/1146 [==============================] - 0s 33us/step
4591/4591 [==============================] - 0s 47us/step
1146/1146 [==============================] - 0s 49us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.25649946306790955,
.9091701154822074]
Current Model Test Results (Loss, Acc.): [0.2788797371986649, 0.
040139609814523]
4591/4591 [==============================] - 0s 48us/step
1146/1146 [==============================] - 0s 45us/step
```

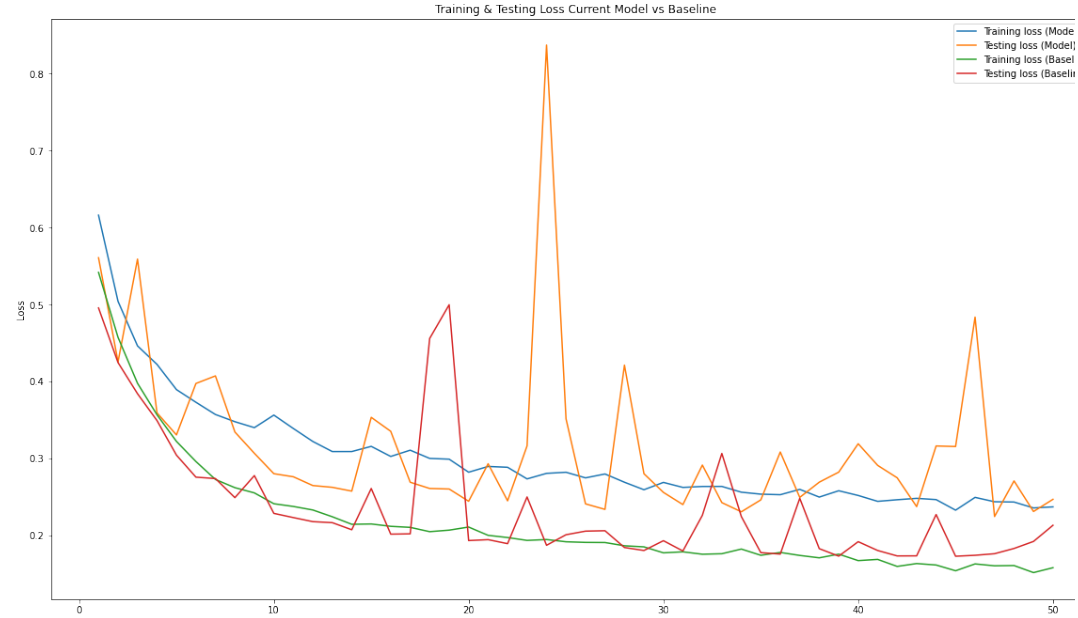




Training & Testing Accuracy Current Model vs Baseline

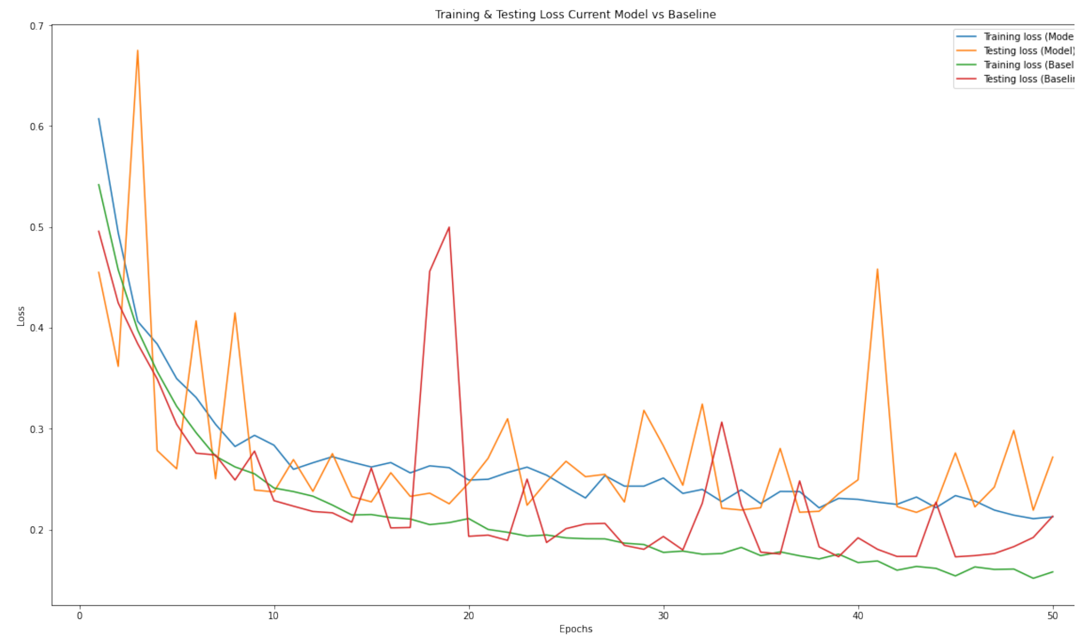


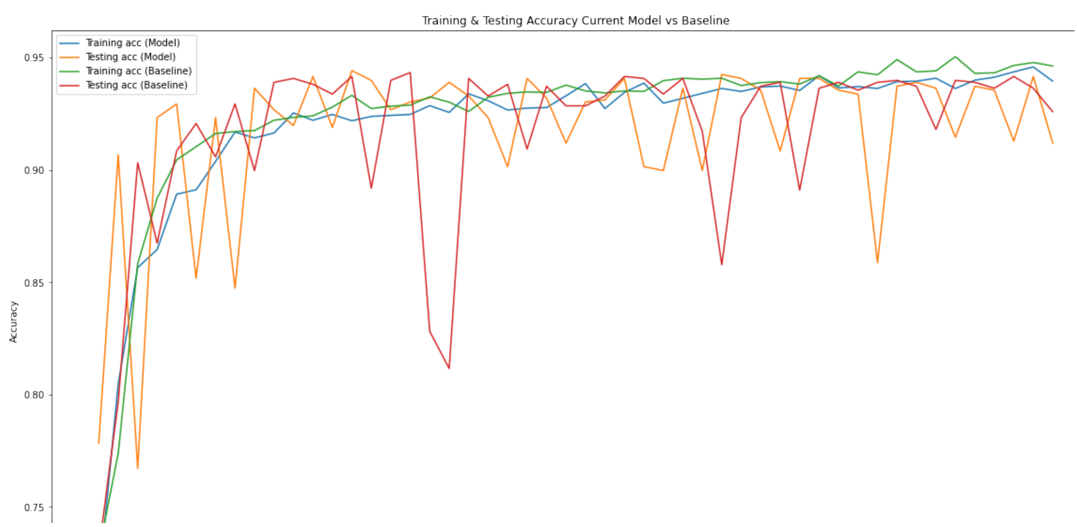
Training & Testing Loss Current Model vs Baseline



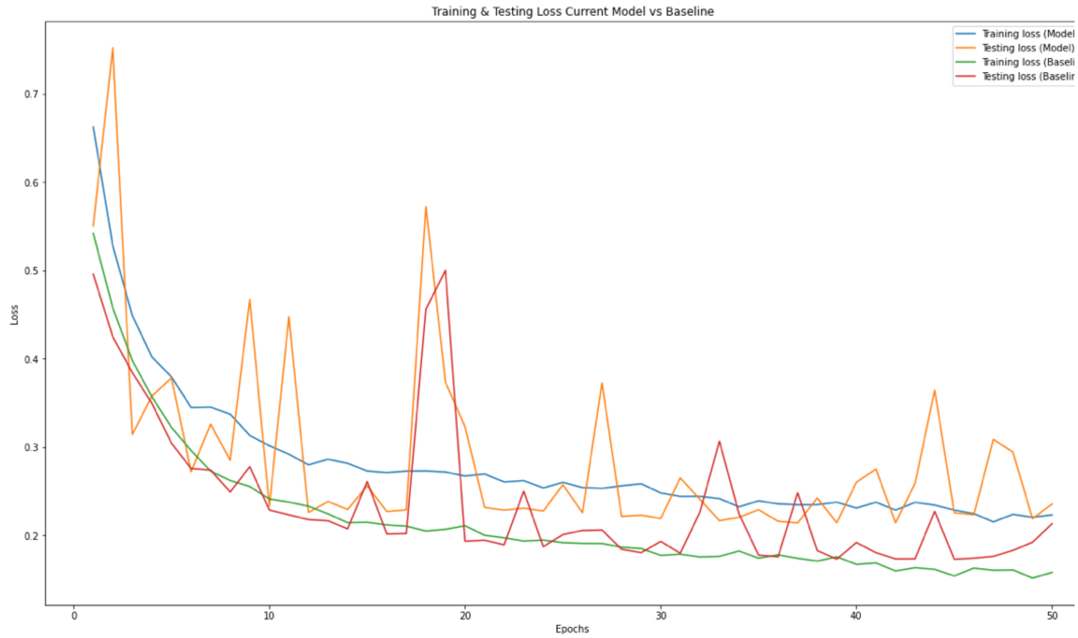
Training & Testing Accuracy Current Model vs Baseline

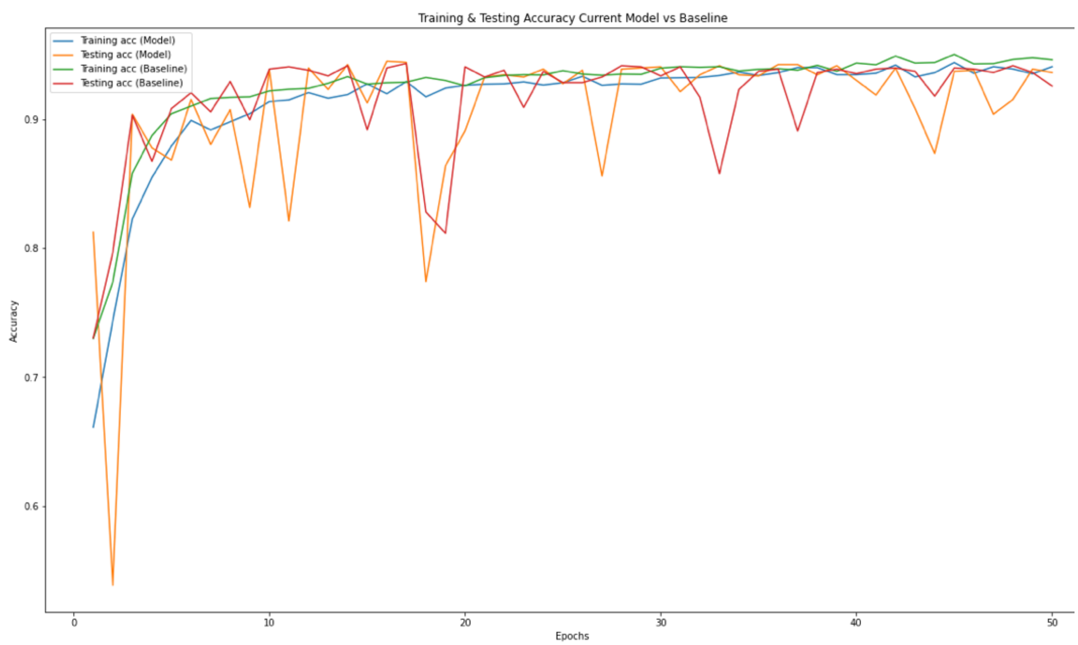

Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy
Epochs

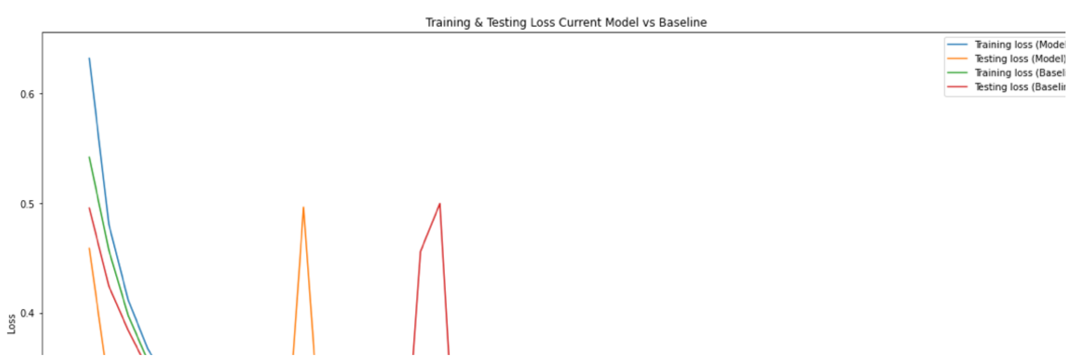

Training & Testing Loss Current Model vs Baseline
Loss
Epochs

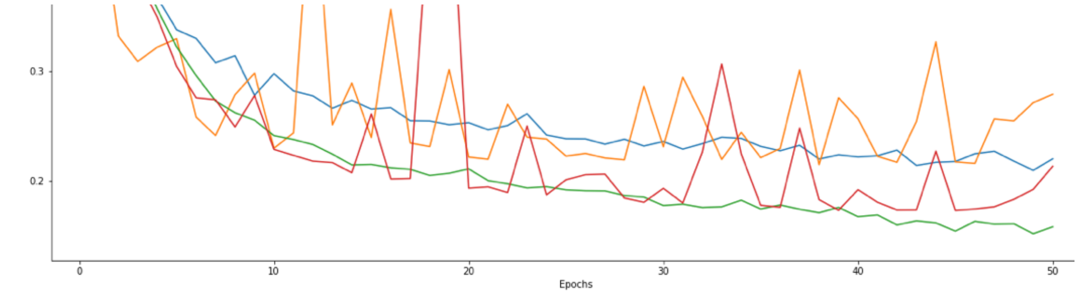

Training & Testing Accuracy Current Model vs Baseline
Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy

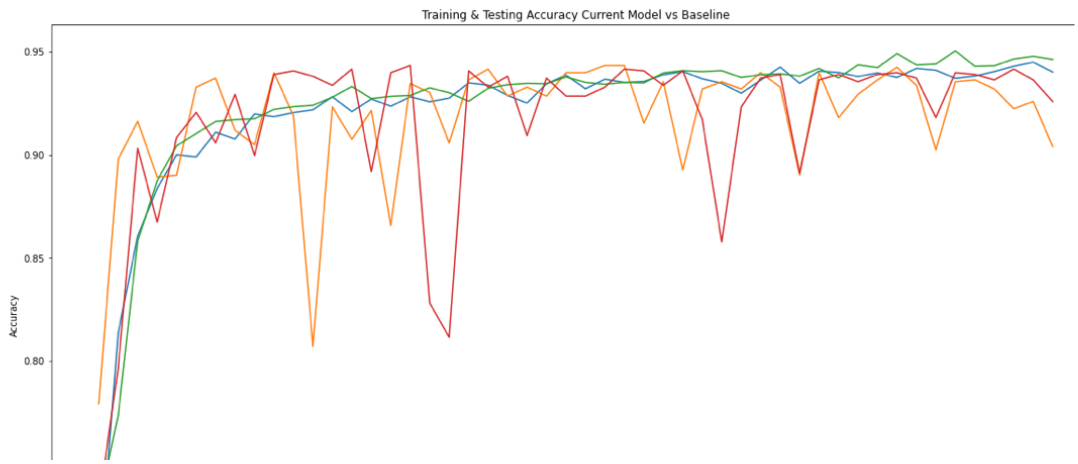

Epochs


Training & Testing Loss Current Model vs Baseline
Training loss (Model
Testing loss (Model)
Training loss (Baseli
Testing loss (Baselir
Loss
Epochs


Training & Testing Accuracy Current Model vs Baseline
Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy
Epochs


Training & Testing Loss Current Model vs Baseline
Training loss (Model
Testing loss (Model)
Training loss (Basel
Testing loss (Baselir
Loss





In [44]:

```
# Printing loss and accuracy for all folds

for item in range(0,len(fold_results_1)):
    print('Fold',item+1,":\n")
    pprint.pprint(fold_results_1[item])
    print('\n')
```

```
Fold 1 :

{'Test Results:': [0.2292993969006064, 0.9363001738959374],
 'Train Results:': [0.18508577658067574, 0.9501197996079286]}


Fold 2 :

{'Test Results:': [0.24686214424032607, 0.9310645718016965],
 'Train Results:': [0.19995980290929313, 0.9488128947941625]}


Fold 3 :

{'Test Results:': [0.27179394040848365, 0.9118673641228134],
 'Train Results:': [0.24837131513616562, 0.9194075365233748]}


Fold 4 :

{'Test Results:': [0.23549430917486885, 0.9363001738959374],
```

```
 'Train Results:': [0.20085735381805048, 0.9431496406011762]}


Fold 5 :

{'Test Results:': [0.2788797371986649, 0.9040139609814523],
 'Train Results:': [0.25649946306790955, 0.9091701154822074]}
```

Comparing our models to the ones above, we are almost fitting perfectly between train and test. Let's now see if we can attempt a CNN model using KFold to generate better results. Please note that this model will take much longer than a traditional neural network, so if the results are not improving dramatically, it is best to stick with a traditional neural network.

Model 2: CNN Model

In [47]:

```python
# Creating one last function to compare CNN model to baseline

def compare_model_results_CNN(baseline_model, baseline_model_fit,

    model_baseline = baseline_model
    model_baseline_fit = baseline_model_fit
    model_func = model
    model_fit_func = model_fit
    results_train_baseline = model_baseline.evaluate(X_train, y_t
    results_test_baseline = model_baseline.evaluate(X_test, y_tes
    results_train_model = model_func.evaluate(train_images, y_tra
    results_test_model = model_func.evaluate(test_images, y_test)

    baseline_model_val_dict = model_baseline_fit.history
    baseline_model_val_dict.keys()
    model_val_dict = model_fit_func.history
    model_val_dict.keys()

    model_loss_values = model_val_dict['loss']
    model_val_loss_values = model_val_dict['val_loss']
    model_acc_values = model_val_dict['acc']
    model_val_acc_values = model_val_dict['val_acc']
    model_epochs = range(1, len(model_loss_values) + 1)

    baseline_loss_values = baseline_model_val_dict['loss']
    baseline_val_loss_values = baseline_model_val_dict['val_loss'
    baseline_acc_values = baseline_model_val_dict['acc']
    baseline_val_acc_values = baseline_model_val_dict['val_acc']
    baseline_epochs = range(1, len(baseline_loss_values) + 1)

    fig, (ax, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(20,

    ax.plot(model_epochs, model_loss_values, label='Training loss
```

```
    ax.plot(model_epochs, model_val_loss_values, label='Testing l
    ax.plot(baseline_epochs, baseline_loss_values, label='Trainin
    ax.plot(baseline_epochs, baseline_val_loss_values, label='Tes
    ax.set_title('Training & Validation Loss Current Model vs Bas
    ax.set_xlabel('Epochs')
    ax.set_ylabel('Loss')
    ax.legend();

    ax2.plot(model_epochs, model_acc_values, label='Training acc
    ax2.plot(model_epochs, model_val_acc_values, label='Testing a
    ax2.plot(baseline_epochs, baseline_acc_values, label='Trainin
    ax2.plot(baseline_epochs, baseline_val_acc_values, label='Tes
    ax2.set_title('Training & Testing Accuracy Current Model vs B
    ax2.set_xlabel('Epochs')
    ax2.set_ylabel('Accuracy')
    ax2.legend();

    return print("\nBaseline Model Train Results (Loss, Acc.):",
```

In [48]:

```
# KFold Cross Validation in conjuction with CNN Model

start = datetime.datetime.now()
kf = KFold(5, shuffle=True, random_state=123)
fold=0
fold_results_2 = []
for i in kf.split(X, Y):
    fold+=1
    print("Results for fold",fold)
    model_2 = models.Sequential()

    # Block 1
    model_2.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(64 ,64,  3)))
    model_2.add(layers.MaxPooling2D((2, 2)))

    #Block 2
    model_2.add(layers.Conv2D(32, (4, 4), activation='relu'))
    model_2.add(layers.MaxPooling2D((2, 2)))

    #Block 3
    model_2.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model_2.add(layers.MaxPooling2D((2, 2)))

    # Dense Block
    model_2.add(layers.Flatten())
    model_2.add(layers.Dense(20, activation='relu', kernel_regula
    model_2.add(layers.Dense(20, activation='relu', kernel_regula
    model_2.add(layers.Dropout(0.5))
    model_2.add(layers.Dense(1, activation='sigmoid'))

    model_2.compile(optimizer='sgd',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
```

```
                 metrics=['accuracy'])

    model_2_fit = model_2.fit(train_images,
                              y_train,
                              epochs=50,
                              batch_size=32,
                              validation_data=(test_images, y_tes

    compare_model_results_CNN(baseline_model,baseline_model_fit,m
    results_2 = {"Train Results:" : model_2.evaluate(train_images
    fold_results_2.append(results_2)
```

```
Results for fold 1
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 16s 3ms/step - loss
0.6851 - acc: 0.7046 - val_loss: 0.6544 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 15s 3ms/step - loss
0.6578 - acc: 0.7286 - val_loss: 0.6469 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.6483 - acc: 0.7299 - val_loss: 0.6373 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6368 - acc: 0.7299 - val_loss: 0.6241 - val_acc: 0.7304
Epoch 5/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6160 - acc: 0.7299 - val_loss: 0.6306 - val_acc: 0.7304
Epoch 6/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.5928 - acc: 0.7299 - val_loss: 0.5598 - val_acc: 0.7304
Epoch 7/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.5728 - acc: 0.7284 - val_loss: 0.5583 - val_acc: 0.7304
Epoch 8/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.5607 - acc: 0.7288 - val_loss: 0.4851 - val_acc: 0.7304
Epoch 9/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.5600 - acc: 0.7277 - val_loss: 0.4629 - val_acc: 0.7304
Epoch 10/50
4591/4591 [==============================] - 15s 3ms/step - loss
0.5265 - acc: 0.7288 - val_loss: 0.4292 - val_acc: 0.7304
Epoch 11/50
4591/4591 [==============================] - 16s 3ms/step - loss
0.4810 - acc: 0.7286 - val_loss: 0.4216 - val_acc: 0.7304
Epoch 12/50
4591/4591 [==============================] - 15s 3ms/step - loss
0.4338 - acc: 0.7735 - val_loss: 0.3764 - val_acc: 0.7304
Epoch 13/50
4591/4591 [==============================] - 15s 3ms/step - loss
0.4052 - acc: 0.8068 - val_loss: 0.3711 - val_acc: 0.9005
Epoch 14/50
```

```
4591/4591 [==============================] - 15s 3ms/step - loss
0.3877 - acc: 0.8338 - val_loss: 0.3518 - val_acc: 0.8988
Epoch 15/50
4591/4591 [==============================] - 15s 3ms/step - loss
0.3760 - acc: 0.8941 - val_loss: 0.3329 - val_acc: 0.9180
Epoch 16/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3549 - acc: 0.9033 - val_loss: 0.3194 - val_acc: 0.9215
Epoch 17/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3466 - acc: 0.9059 - val_loss: 0.4056 - val_acc: 0.8839
Epoch 18/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3580 - acc: 0.8994 - val_loss: 0.3031 - val_acc: 0.9223
Epoch 19/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3290 - acc: 0.9094 - val_loss: 0.3451 - val_acc: 0.8761
Epoch 20/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3301 - acc: 0.9142 - val_loss: 0.3332 - val_acc: 0.8866
Epoch 21/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3237 - acc: 0.9124 - val_loss: 0.2798 - val_acc: 0.9293
Epoch 22/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3127 - acc: 0.9168 - val_loss: 0.2755 - val_acc: 0.9267
Epoch 23/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3104 - acc: 0.9161 - val_loss: 0.3612 - val_acc: 0.8586
Epoch 24/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3028 - acc: 0.9151 - val_loss: 0.2754 - val_acc: 0.9293
Epoch 25/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2998 - acc: 0.9179 - val_loss: 0.3604 - val_acc: 0.8988
Epoch 26/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2993 - acc: 0.9203 - val_loss: 0.2681 - val_acc: 0.9319
Epoch 27/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2988 - acc: 0.9214 - val_loss: 0.3262 - val_acc: 0.8770
Epoch 28/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.3026 - acc: 0.9190 - val_loss: 0.2550 - val_acc: 0.9354
Epoch 29/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3013 - acc: 0.9212 - val_loss: 0.2513 - val_acc: 0.9389
Epoch 30/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2900 - acc: 0.9251 - val_loss: 0.2727 - val_acc: 0.9319
Epoch 31/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2915 - acc: 0.9257 - val_loss: 0.3961 - val_acc: 0.8787
Epoch 32/50
```

```
Epoch 32/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2834 - acc: 0.9238 - val_loss: 0.2513 - val_acc: 0.9284
Epoch 33/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2826 - acc: 0.9305 - val_loss: 0.4107 - val_acc: 0.8813
Epoch 34/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2817 - acc: 0.9259 - val_loss: 0.2411 - val_acc: 0.9389
Epoch 35/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2745 - acc: 0.9264 - val_loss: 0.3064 - val_acc: 0.9241
Epoch 36/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2762 - acc: 0.9303 - val_loss: 0.4578 - val_acc: 0.8150
Epoch 37/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2708 - acc: 0.9320 - val_loss: 0.2476 - val_acc: 0.9302
Epoch 38/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2827 - acc: 0.9277 - val_loss: 0.2920 - val_acc: 0.9267
Epoch 39/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2673 - acc: 0.9323 - val_loss: 0.2814 - val_acc: 0.9119
Epoch 40/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2652 - acc: 0.9323 - val_loss: 0.3244 - val_acc: 0.8831
Epoch 41/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2601 - acc: 0.9333 - val_loss: 0.2314 - val_acc: 0.9415
Epoch 42/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2609 - acc: 0.9342 - val_loss: 0.2252 - val_acc: 0.9442
Epoch 43/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2551 - acc: 0.9331 - val_loss: 0.2243 - val_acc: 0.9442
Epoch 44/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2551 - acc: 0.9347 - val_loss: 0.2617 - val_acc: 0.9346
Epoch 45/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2539 - acc: 0.9364 - val_loss: 0.2833 - val_acc: 0.9232
Epoch 46/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2508 - acc: 0.9371 - val_loss: 0.2462 - val_acc: 0.9372
Epoch 47/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2491 - acc: 0.9351 - val_loss: 0.2318 - val_acc: 0.9433
Epoch 48/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2513 - acc: 0.9357 - val_loss: 0.2163 - val_acc: 0.9442
Epoch 49/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2474 - acc: 0.9366 - val_loss: 0.7449 - val_acc: 0.6545
```

```
Epoch 50/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2678 - acc: 0.9288 - val_loss: 0.2165 - val_acc: 0.9442
4591/4591 [==============================] - 0s 30us/step
1146/1146 [==============================] - 0s 27us/step
4591/4591 [==============================] - 4s 801us/step
1146/1146 [==============================] - 1s 808us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.20279987057071708,
.9509910694837725]
Current Model Test Results (Loss, Acc.): [0.21654142855453656, 0
9441535770372986]
4591/4591 [==============================] - 4s 795us/step
1146/1146 [==============================] - 1s 802us/step
Results for fold 2
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6820 - acc: 0.7099 - val_loss: 0.6420 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6458 - acc: 0.7293 - val_loss: 0.6359 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6242 - acc: 0.7330 - val_loss: 0.5943 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.5881 - acc: 0.7478 - val_loss: 0.4985 - val_acc: 0.8351
Epoch 5/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.5426 - acc: 0.7722 - val_loss: 0.4659 - val_acc: 0.7766
Epoch 6/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.4740 - acc: 0.8079 - val_loss: 0.4159 - val_acc: 0.8255
Epoch 7/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.4233 - acc: 0.8421 - val_loss: 0.3170 - val_acc: 0.8901
Epoch 8/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3630 - acc: 0.8763 - val_loss: 0.2939 - val_acc: 0.9049
Epoch 9/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3473 - acc: 0.8863 - val_loss: 0.2805 - val_acc: 0.9127
Epoch 10/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3211 - acc: 0.9000 - val_loss: 0.2649 - val_acc: 0.9066
Epoch 11/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3028 - acc: 0.9031 - val_loss: 0.3546 - val_acc: 0.8805
```

```
Epoch 12/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2925 - acc: 0.9066 - val_loss: 0.2550 - val_acc: 0.9284
Epoch 13/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2899 - acc: 0.9140 - val_loss: 0.2506 - val_acc: 0.9302
Epoch 14/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2790 - acc: 0.9155 - val_loss: 0.2556 - val_acc: 0.9154
Epoch 15/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2753 - acc: 0.9129 - val_loss: 0.2394 - val_acc: 0.9346
Epoch 16/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2733 - acc: 0.9198 - val_loss: 0.2478 - val_acc: 0.9293
Epoch 17/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2682 - acc: 0.9198 - val_loss: 0.2273 - val_acc: 0.9346
Epoch 18/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2572 - acc: 0.9222 - val_loss: 0.2241 - val_acc: 0.9372
Epoch 19/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2663 - acc: 0.9231 - val_loss: 0.2319 - val_acc: 0.9328
Epoch 20/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2577 - acc: 0.9251 - val_loss: 0.2221 - val_acc: 0.9354
Epoch 21/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2507 - acc: 0.9229 - val_loss: 0.3121 - val_acc: 0.8909
Epoch 22/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2550 - acc: 0.9281 - val_loss: 0.2261 - val_acc: 0.9398
Epoch 23/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2474 - acc: 0.9288 - val_loss: 0.2116 - val_acc: 0.9424
Epoch 24/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2444 - acc: 0.9288 - val_loss: 0.2126 - val_acc: 0.9415
Epoch 25/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2414 - acc: 0.9325 - val_loss: 0.2103 - val_acc: 0.9398
Epoch 26/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2394 - acc: 0.9290 - val_loss: 0.2028 - val_acc: 0.9433
Epoch 27/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2399 - acc: 0.9325 - val_loss: 0.2333 - val_acc: 0.9319
Epoch 28/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2319 - acc: 0.9353 - val_loss: 0.2558 - val_acc: 0.9293
Epoch 29/50
4591/4591 [==============================] - 13s 3ms/step - loss
```

```
0.2351 - acc: 0.9340 - val_loss: 0.2132 - val_acc: 0.9363
Epoch 30/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2288 - acc: 0.9366 - val_loss: 0.2587 - val_acc: 0.9232
Epoch 31/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2381 - acc: 0.9318 - val_loss: 0.2035 - val_acc: 0.9442
Epoch 32/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2313 - acc: 0.9357 - val_loss: 0.1943 - val_acc: 0.9485
Epoch 33/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2258 - acc: 0.9377 - val_loss: 0.2062 - val_acc: 0.9433
Epoch 34/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2177 - acc: 0.9412 - val_loss: 0.2184 - val_acc: 0.9372
Epoch 35/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2271 - acc: 0.9390 - val_loss: 0.2422 - val_acc: 0.9197
Epoch 36/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2260 - acc: 0.9379 - val_loss: 0.1879 - val_acc: 0.9529
Epoch 37/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2217 - acc: 0.9388 - val_loss: 0.2071 - val_acc: 0.9485
Epoch 38/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2178 - acc: 0.9416 - val_loss: 0.1903 - val_acc: 0.9555
Epoch 39/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2133 - acc: 0.9418 - val_loss: 0.1877 - val_acc: 0.9555
Epoch 40/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2186 - acc: 0.9379 - val_loss: 0.2199 - val_acc: 0.9337
Epoch 41/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2141 - acc: 0.9421 - val_loss: 0.2288 - val_acc: 0.9337
Epoch 42/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2047 - acc: 0.9445 - val_loss: 0.1859 - val_acc: 0.9511
Epoch 43/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2036 - acc: 0.9418 - val_loss: 0.3047 - val_acc: 0.8953
Epoch 44/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.2018 - acc: 0.9495 - val_loss: 0.2572 - val_acc: 0.9267
Epoch 45/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2039 - acc: 0.9421 - val_loss: 0.1916 - val_acc: 0.9529
Epoch 46/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2068 - acc: 0.9397 - val_loss: 0.1825 - val_acc: 0.9476
Epoch 47/50
4591/4591 [==============================] - 12s 3ms/step - loss
```

```
0.1989 - acc: 0.9431 - val_loss: 0.1869 - val_acc: 0.9538
Epoch 48/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1929 - acc: 0.9464 - val_loss: 0.1926 - val_acc: 0.9433
Epoch 49/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2014 - acc: 0.9475 - val_loss: 0.1773 - val_acc: 0.9520
Epoch 50/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2071 - acc: 0.9458 - val_loss: 0.2289 - val_acc: 0.9380
4591/4591 [==============================] - 0s 30us/step
1146/1146 [==============================] - 0s 28us/step
4591/4591 [==============================] - 3s 731us/step
1146/1146 [==============================] - 1s 735us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.2202661924179383, 0
9326944020910477]
Current Model Test Results (Loss, Acc.): [0.22888071015867265, 0
9380453745940177]
4591/4591 [==============================] - 3s 750us/step
1146/1146 [==============================] - 1s 747us/step
Results for fold 3
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.6697 - acc: 0.7286 - val_loss: 0.6454 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6505 - acc: 0.7299 - val_loss: 0.6352 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6215 - acc: 0.7303 - val_loss: 0.5819 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5633 - acc: 0.7462 - val_loss: 0.4670 - val_acc: 0.7827
Epoch 5/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5082 - acc: 0.7665 - val_loss: 0.4043 - val_acc: 0.8682
Epoch 6/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.4707 - acc: 0.7822 - val_loss: 0.4778 - val_acc: 0.7373
Epoch 7/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.4018 - acc: 0.8408 - val_loss: 0.6063 - val_acc: 0.7993
Epoch 8/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.3792 - acc: 0.8567 - val_loss: 0.4076 - val_acc: 0.8202
Epoch 9/50
```

```
4591/4591 [==============================] - 12s 3ms/step - loss
0.3522 - acc: 0.8787 - val_loss: 0.3592 - val_acc: 0.8831
Epoch 10/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3276 - acc: 0.8915 - val_loss: 0.4671 - val_acc: 0.7827
Epoch 11/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3110 - acc: 0.8944 - val_loss: 0.3242 - val_acc: 0.8988
Epoch 12/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3116 - acc: 0.9013 - val_loss: 0.2817 - val_acc: 0.9136
Epoch 13/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2957 - acc: 0.9107 - val_loss: 0.3406 - val_acc: 0.8534
Epoch 14/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2906 - acc: 0.9103 - val_loss: 0.2600 - val_acc: 0.9284
Epoch 15/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2807 - acc: 0.9155 - val_loss: 0.2719 - val_acc: 0.9250
Epoch 16/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2680 - acc: 0.9212 - val_loss: 0.3743 - val_acc: 0.8290
Epoch 17/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2654 - acc: 0.9212 - val_loss: 0.2440 - val_acc: 0.9346
Epoch 18/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2664 - acc: 0.9196 - val_loss: 0.2286 - val_acc: 0.9363
Epoch 19/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2588 - acc: 0.9270 - val_loss: 0.2225 - val_acc: 0.9389
Epoch 20/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2535 - acc: 0.9235 - val_loss: 0.2160 - val_acc: 0.9424
Epoch 21/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2537 - acc: 0.9329 - val_loss: 0.2607 - val_acc: 0.9346
Epoch 22/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2526 - acc: 0.9270 - val_loss: 0.2164 - val_acc: 0.9407
Epoch 23/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2377 - acc: 0.9399 - val_loss: 0.2121 - val_acc: 0.9372
Epoch 24/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2387 - acc: 0.9410 - val_loss: 0.2293 - val_acc: 0.9389
Epoch 25/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2373 - acc: 0.9331 - val_loss: 0.3168 - val_acc: 0.9171
Epoch 26/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2344 - acc: 0.9371 - val_loss: 0.2467 - val_acc: 0.9337
Epoch 27/50
```

```
4591/4591 [==============================] - 12s 3ms/step - loss
0.2286 - acc: 0.9401 - val_loss: 0.2045 - val_acc: 0.9503
Epoch 28/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2327 - acc: 0.9360 - val_loss: 0.2164 - val_acc: 0.9459
Epoch 29/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2294 - acc: 0.9386 - val_loss: 0.3395 - val_acc: 0.8665
Epoch 30/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2250 - acc: 0.9388 - val_loss: 0.2052 - val_acc: 0.9372
Epoch 31/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2173 - acc: 0.9412 - val_loss: 0.2119 - val_acc: 0.9485
Epoch 32/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2124 - acc: 0.9466 - val_loss: 0.1954 - val_acc: 0.9511
Epoch 33/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2110 - acc: 0.9436 - val_loss: 0.2116 - val_acc: 0.9476
Epoch 34/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2189 - acc: 0.9440 - val_loss: 0.2044 - val_acc: 0.9476
Epoch 35/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2129 - acc: 0.9469 - val_loss: 0.2477 - val_acc: 0.9188
Epoch 36/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2130 - acc: 0.9425 - val_loss: 0.1975 - val_acc: 0.9529
Epoch 37/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2055 - acc: 0.9506 - val_loss: 0.1928 - val_acc: 0.9433
Epoch 38/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2121 - acc: 0.9445 - val_loss: 0.1876 - val_acc: 0.9494
Epoch 39/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2063 - acc: 0.9479 - val_loss: 0.2475 - val_acc: 0.9293
Epoch 40/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2021 - acc: 0.9497 - val_loss: 0.1966 - val_acc: 0.9564
Epoch 41/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2006 - acc: 0.9471 - val_loss: 0.2152 - val_acc: 0.9407
Epoch 42/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2036 - acc: 0.9506 - val_loss: 0.1750 - val_acc: 0.9546
Epoch 43/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1937 - acc: 0.9547 - val_loss: 0.2028 - val_acc: 0.9433
Epoch 44/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1967 - acc: 0.9527 - val_loss: 0.2522 - val_acc: 0.9267
Epoch 45/50
```

```
Epoch 45/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2011 - acc: 0.9510 - val_loss: 0.1751 - val_acc: 0.9546
Epoch 46/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1963 - acc: 0.9519 - val_loss: 0.2163 - val_acc: 0.9337
Epoch 47/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1874 - acc: 0.9508 - val_loss: 0.1720 - val_acc: 0.9572
Epoch 48/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1913 - acc: 0.9553 - val_loss: 0.2158 - val_acc: 0.9468
Epoch 49/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1823 - acc: 0.9543 - val_loss: 0.1739 - val_acc: 0.9590
Epoch 50/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1893 - acc: 0.9545 - val_loss: 0.1728 - val_acc: 0.9494
4591/4591 [==============================] - 0s 29us/step
1146/1146 [==============================] - 0s 27us/step
4591/4591 [==============================] - 3s 702us/step
1146/1146 [==============================] - 1s 712us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.15459524097330454,
.9623175778827281]
Current Model Test Results (Loss, Acc.): [0.17284238736354868, 0
9493891791315395]
4591/4591 [==============================] - 3s 692us/step
1146/1146 [==============================] - 1s 709us/step
Results for fold 4
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6691 - acc: 0.7275 - val_loss: 0.6494 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6403 - acc: 0.7301 - val_loss: 0.6200 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5977 - acc: 0.7371 - val_loss: 0.5413 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5617 - acc: 0.7656 - val_loss: 0.7622 - val_acc: 0.4834
Epoch 5/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5025 - acc: 0.7937 - val_loss: 0.4358 - val_acc: 0.8630
Epoch 6/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.4326 - acc: 0.8360 - val_loss: 0.3277 - val_acc: 0.8970
```

```
Epoch 7/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3852 - acc: 0.8593 - val_loss: 0.3112 - val_acc: 0.8935
Epoch 8/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3499 - acc: 0.8767 - val_loss: 0.3810 - val_acc: 0.8639
Epoch 9/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3276 - acc: 0.8952 - val_loss: 0.2603 - val_acc: 0.9171
Epoch 10/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3206 - acc: 0.8981 - val_loss: 0.2585 - val_acc: 0.9223
Epoch 11/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3026 - acc: 0.9022 - val_loss: 0.2481 - val_acc: 0.9311
Epoch 12/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3027 - acc: 0.9068 - val_loss: 0.2580 - val_acc: 0.9241
Epoch 13/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2839 - acc: 0.9168 - val_loss: 0.2376 - val_acc: 0.9319
Epoch 14/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2834 - acc: 0.9157 - val_loss: 0.2595 - val_acc: 0.9241
Epoch 15/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2851 - acc: 0.9127 - val_loss: 0.2310 - val_acc: 0.9346
Epoch 16/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2690 - acc: 0.9253 - val_loss: 0.2301 - val_acc: 0.9346
Epoch 17/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2644 - acc: 0.9220 - val_loss: 0.2270 - val_acc: 0.9354
Epoch 18/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2720 - acc: 0.9259 - val_loss: 0.2920 - val_acc: 0.9040
Epoch 19/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2599 - acc: 0.9242 - val_loss: 0.3229 - val_acc: 0.8831
Epoch 20/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2554 - acc: 0.9275 - val_loss: 0.3143 - val_acc: 0.8962
Epoch 21/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2567 - acc: 0.9246 - val_loss: 0.2152 - val_acc: 0.9398
Epoch 22/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2573 - acc: 0.9294 - val_loss: 0.2424 - val_acc: 0.9372
Epoch 23/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2518 - acc: 0.9266 - val_loss: 0.2771 - val_acc: 0.9127
Epoch 24/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2525 - acc: 0.9299 - val_loss: 0.2237 - val_acc: 0.9328
```

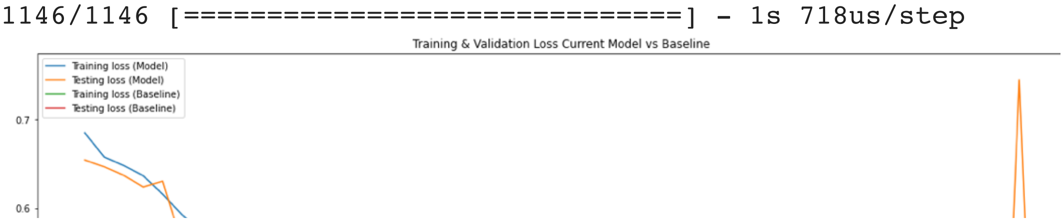
```
Epoch 25/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2564 - acc: 0.9259 - val_loss: 0.2215 - val_acc: 0.9372
Epoch 26/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2407 - acc: 0.9351 - val_loss: 0.2530 - val_acc: 0.9206
Epoch 27/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2420 - acc: 0.9312 - val_loss: 0.2176 - val_acc: 0.9415
Epoch 28/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2381 - acc: 0.9340 - val_loss: 0.2275 - val_acc: 0.9337
Epoch 29/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2343 - acc: 0.9384 - val_loss: 0.2002 - val_acc: 0.9433
Epoch 30/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2337 - acc: 0.9360 - val_loss: 0.2010 - val_acc: 0.9398
Epoch 31/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2365 - acc: 0.9366 - val_loss: 0.1970 - val_acc: 0.9442
Epoch 32/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2348 - acc: 0.9347 - val_loss: 0.2205 - val_acc: 0.9433
Epoch 33/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2303 - acc: 0.9390 - val_loss: 0.2068 - val_acc: 0.9415
Epoch 34/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2236 - acc: 0.9353 - val_loss: 0.2014 - val_acc: 0.9476
Epoch 35/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2244 - acc: 0.9384 - val_loss: 0.1922 - val_acc: 0.9468
Epoch 36/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2293 - acc: 0.9375 - val_loss: 0.3175 - val_acc: 0.9014
Epoch 37/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2184 - acc: 0.9410 - val_loss: 0.3606 - val_acc: 0.8630
Epoch 38/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2111 - acc: 0.9429 - val_loss: 0.1954 - val_acc: 0.9442
Epoch 39/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2140 - acc: 0.9451 - val_loss: 0.1858 - val_acc: 0.9468
Epoch 40/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2090 - acc: 0.9469 - val_loss: 0.1854 - val_acc: 0.9529
Epoch 41/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2105 - acc: 0.9399 - val_loss: 0.1846 - val_acc: 0.9494
Epoch 42/50
4591/4591 [==============================] - 12s 3ms/step - loss
```

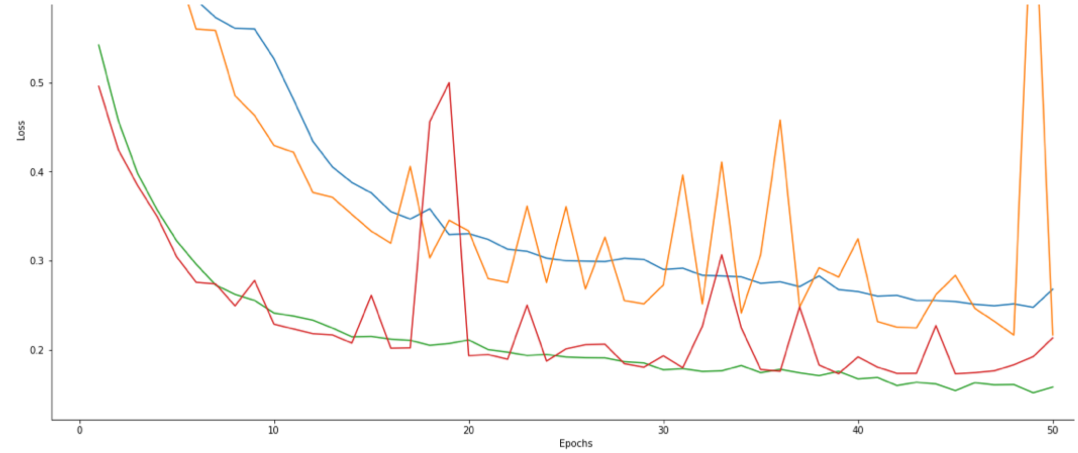
```
0.2048 - acc: 0.9466 - val_loss: 0.1761 - val_acc: 0.9520
Epoch 43/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2088 - acc: 0.9455 - val_loss: 0.2792 - val_acc: 0.8927
Epoch 44/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2018 - acc: 0.9440 - val_loss: 0.2658 - val_acc: 0.9328
Epoch 45/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2007 - acc: 0.9438 - val_loss: 0.2032 - val_acc: 0.9450
Epoch 46/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2001 - acc: 0.9462 - val_loss: 0.1978 - val_acc: 0.9520
Epoch 47/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2005 - acc: 0.9458 - val_loss: 0.1703 - val_acc: 0.9520
Epoch 48/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2001 - acc: 0.9427 - val_loss: 0.1968 - val_acc: 0.9503
Epoch 49/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1956 - acc: 0.9484 - val_loss: 0.1952 - val_acc: 0.9459
Epoch 50/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.1984 - acc: 0.9464 - val_loss: 0.1800 - val_acc: 0.9520
4591/4591 [==============================] - 0s 30us/step
1146/1146 [==============================] - 0s 30us/step
4591/4591 [==============================] - 3s 694us/step
1146/1146 [==============================] - 1s 722us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.16929529276880348,
.9566543236767588]
Current Model Test Results (Loss, Acc.): [0.1799898976638887, 0.
520069801786599]
4591/4591 [==============================] - 3s 705us/step
1146/1146 [==============================] - 1s 714us/step
Results for fold 5
Train on 4591 samples, validate on 1146 samples
Epoch 1/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6637 - acc: 0.7290 - val_loss: 0.6436 - val_acc: 0.7304
Epoch 2/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.6324 - acc: 0.7297 - val_loss: 0.6025 - val_acc: 0.7304
Epoch 3/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5738 - acc: 0.7380 - val_loss: 0.5510 - val_acc: 0.7304
Epoch 4/50
4591/4591 [==============================] - 12s 3ms/step - loss
```

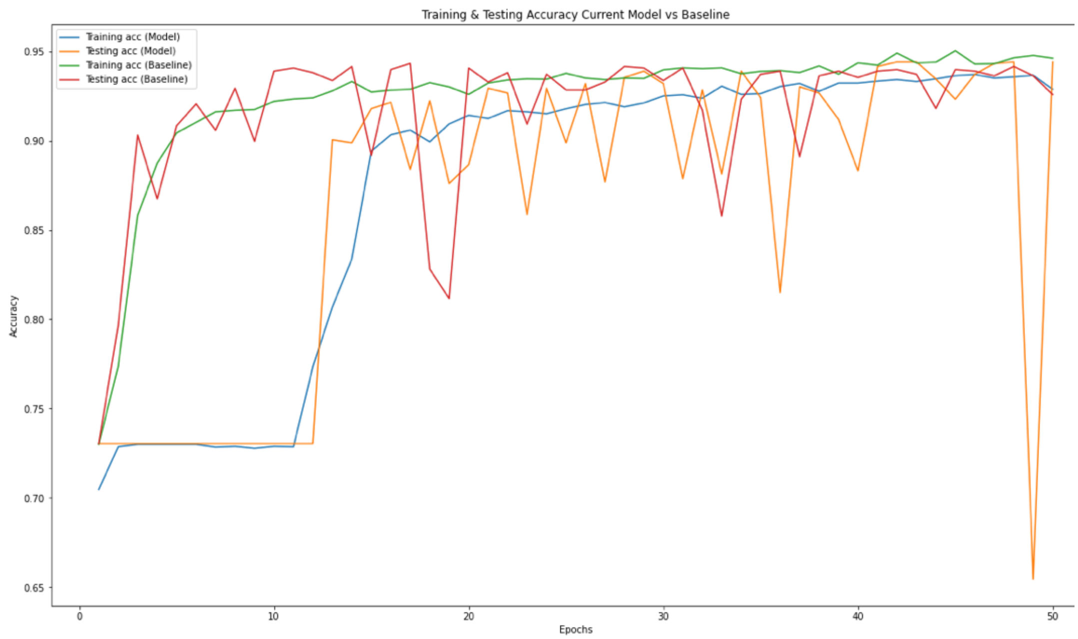
```
0.5516 - acc: 0.7526 - val_loss: 0.4466 - val_acc: 0.7618
Epoch 5/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.5117 - acc: 0.7724 - val_loss: 0.3997 - val_acc: 0.8447
Epoch 6/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.4597 - acc: 0.8022 - val_loss: 0.3844 - val_acc: 0.8796
Epoch 7/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.4203 - acc: 0.8214 - val_loss: 0.3448 - val_acc: 0.8770
Epoch 8/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3743 - acc: 0.8599 - val_loss: 0.3520 - val_acc: 0.8613
Epoch 9/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3568 - acc: 0.8623 - val_loss: 0.2997 - val_acc: 0.9075
Epoch 10/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3272 - acc: 0.8819 - val_loss: 0.2753 - val_acc: 0.8997
Epoch 11/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3237 - acc: 0.8748 - val_loss: 0.3027 - val_acc: 0.9110
Epoch 12/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.3113 - acc: 0.8917 - val_loss: 0.2621 - val_acc: 0.9188
Epoch 13/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2967 - acc: 0.8946 - val_loss: 0.2791 - val_acc: 0.9215
Epoch 14/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2920 - acc: 0.8944 - val_loss: 0.2587 - val_acc: 0.9136
Epoch 15/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2915 - acc: 0.9039 - val_loss: 0.3036 - val_acc: 0.9058
Epoch 16/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2903 - acc: 0.9046 - val_loss: 0.2453 - val_acc: 0.9267
Epoch 17/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2776 - acc: 0.9113 - val_loss: 0.3025 - val_acc: 0.8778
Epoch 18/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2792 - acc: 0.9133 - val_loss: 0.2580 - val_acc: 0.9284
Epoch 19/50
4591/4591 [==============================] - 14s 3ms/step - loss
0.2790 - acc: 0.9194 - val_loss: 0.2596 - val_acc: 0.9311
Epoch 20/50
4591/4591 [==============================] - 13s 3ms/step - loss
0.2741 - acc: 0.9249 - val_loss: 0.2576 - val_acc: 0.9328
Epoch 21/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2668 - acc: 0.9262 - val_loss: 0.2406 - val_acc: 0.9302
Epoch 22/50
```

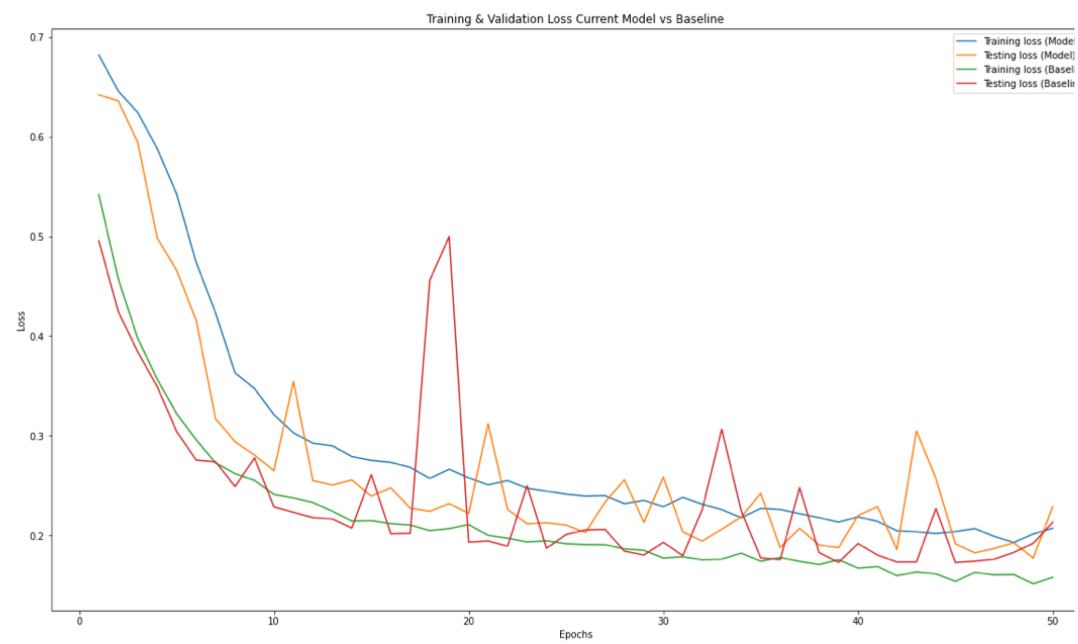
```
4591/4591 [==============================] - 12s 3ms/step - loss
0.2711 - acc: 0.9264 - val_loss: 0.2354 - val_acc: 0.9337
Epoch 23/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2618 - acc: 0.9301 - val_loss: 0.2274 - val_acc: 0.9346
Epoch 24/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2540 - acc: 0.9310 - val_loss: 0.2698 - val_acc: 0.9354
Epoch 25/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2588 - acc: 0.9325 - val_loss: 0.3323 - val_acc: 0.8613
Epoch 26/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2502 - acc: 0.9314 - val_loss: 0.2177 - val_acc: 0.9363
Epoch 27/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2397 - acc: 0.9333 - val_loss: 0.2261 - val_acc: 0.9328
Epoch 28/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2509 - acc: 0.9275 - val_loss: 0.2137 - val_acc: 0.9389
Epoch 29/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2508 - acc: 0.9355 - val_loss: 0.2114 - val_acc: 0.9407
Epoch 30/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2365 - acc: 0.9349 - val_loss: 0.2344 - val_acc: 0.9424
Epoch 31/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2344 - acc: 0.9388 - val_loss: 0.3168 - val_acc: 0.8717
Epoch 32/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2410 - acc: 0.9349 - val_loss: 0.2264 - val_acc: 0.9250
Epoch 33/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2417 - acc: 0.9333 - val_loss: 0.2124 - val_acc: 0.9398
Epoch 34/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2287 - acc: 0.9397 - val_loss: 0.2046 - val_acc: 0.9433
Epoch 35/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2311 - acc: 0.9410 - val_loss: 0.2055 - val_acc: 0.9468
Epoch 36/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2272 - acc: 0.9416 - val_loss: 0.2009 - val_acc: 0.9468
Epoch 37/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2250 - acc: 0.9447 - val_loss: 0.2467 - val_acc: 0.9319
Epoch 38/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2186 - acc: 0.9421 - val_loss: 0.1959 - val_acc: 0.9442
Epoch 39/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2181 - acc: 0.9436 - val_loss: 0.2021 - val_acc: 0.9450
Epoch 40/50
```

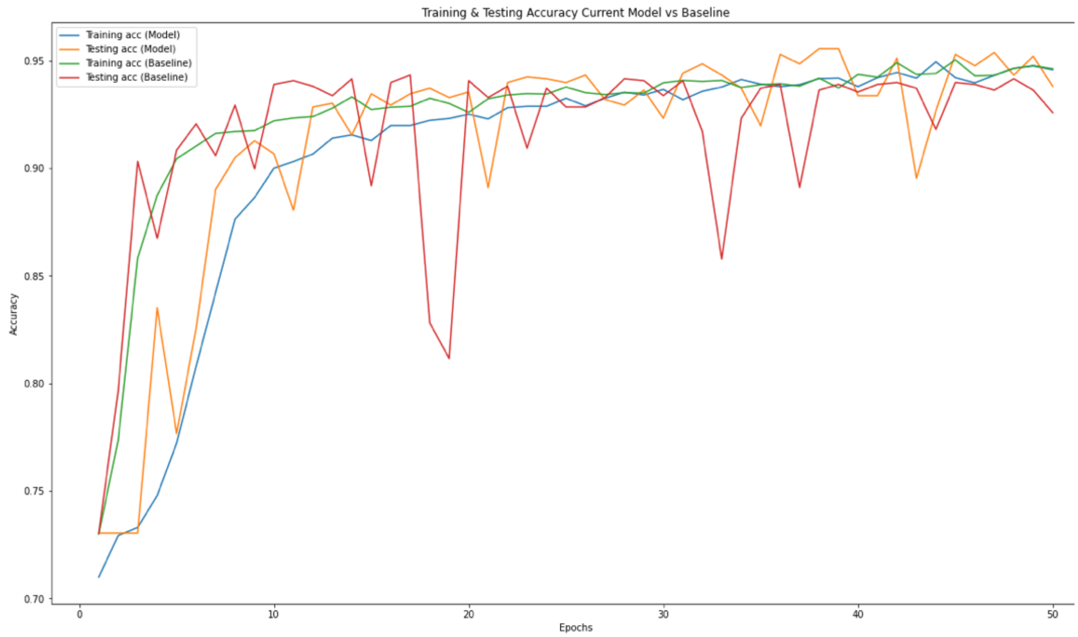
```
4591/4591 [==============================] - 12s 3ms/step - loss
0.2276 - acc: 0.9405 - val_loss: 0.1999 - val_acc: 0.9476
Epoch 41/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2145 - acc: 0.9401 - val_loss: 0.2048 - val_acc: 0.9459
Epoch 42/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2154 - acc: 0.9455 - val_loss: 0.2207 - val_acc: 0.9389
Epoch 43/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2052 - acc: 0.9458 - val_loss: 0.2053 - val_acc: 0.9494
Epoch 44/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2123 - acc: 0.9458 - val_loss: 0.1935 - val_acc: 0.9494
Epoch 45/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2211 - acc: 0.9458 - val_loss: 0.2039 - val_acc: 0.9450
Epoch 46/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2107 - acc: 0.9497 - val_loss: 0.2028 - val_acc: 0.9468
Epoch 47/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2125 - acc: 0.9436 - val_loss: 0.1975 - val_acc: 0.9476
Epoch 48/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2083 - acc: 0.9455 - val_loss: 0.1853 - val_acc: 0.9511
Epoch 49/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2009 - acc: 0.9486 - val_loss: 0.2168 - val_acc: 0.9433
Epoch 50/50
4591/4591 [==============================] - 12s 3ms/step - loss
0.2099 - acc: 0.9429 - val_loss: 0.1964 - val_acc: 0.9511
4591/4591 [==============================] - 0s 29us/step
1146/1146 [==============================] - 0s 31us/step
4591/4591 [==============================] - 3s 716us/step
1146/1146 [==============================] - 1s 696us/step

Baseline Model Train Results (Loss, Acc.): [0.17525029354620228,
0.9313874972902646]
Baseline Model Test Results (Loss, Acc.): [0.2131096540607291, 0
9258289703315882]

Current Model Train Results (Loss, Acc.): [0.17017031406309838,
.9509910694837725]
Current Model Test Results (Loss, Acc.): [0.19644602792529328, 0
9511343798296198]
4591/4591 [==============================] - 3s 702us/step
1146/1146 [==============================] - 1s 718us/step
```

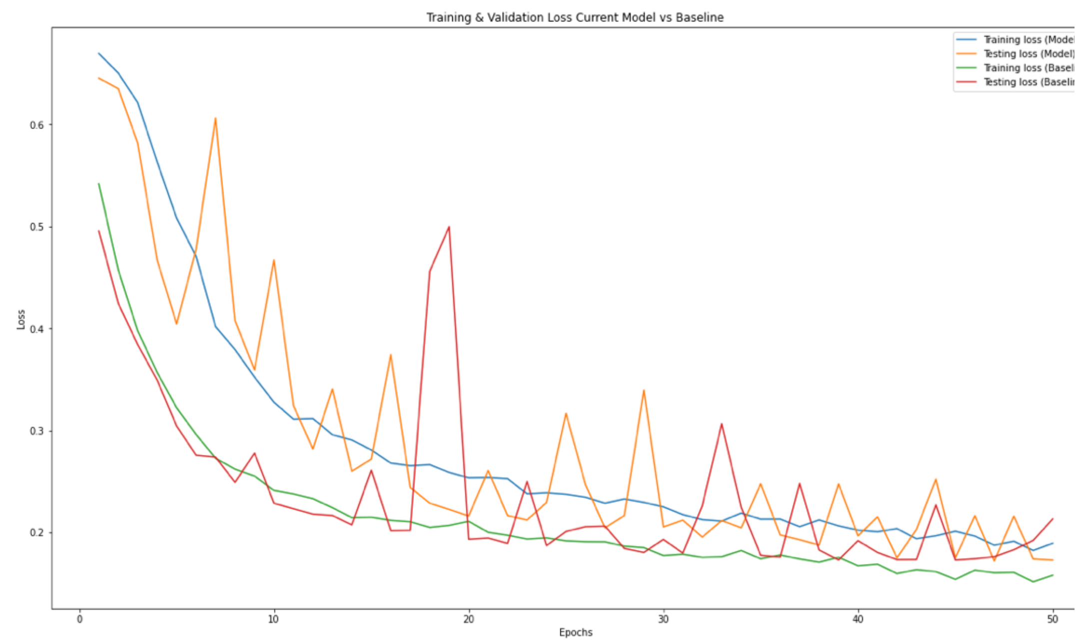


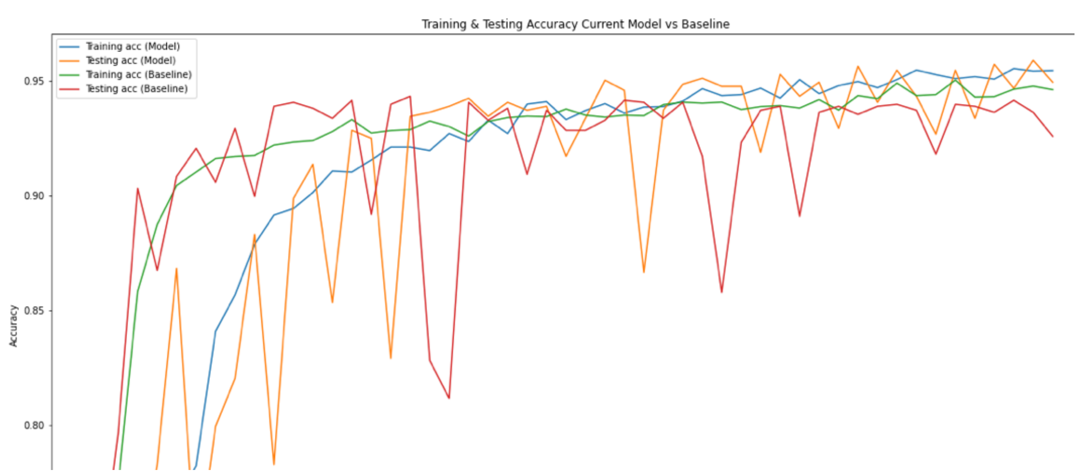

Loss
Epochs

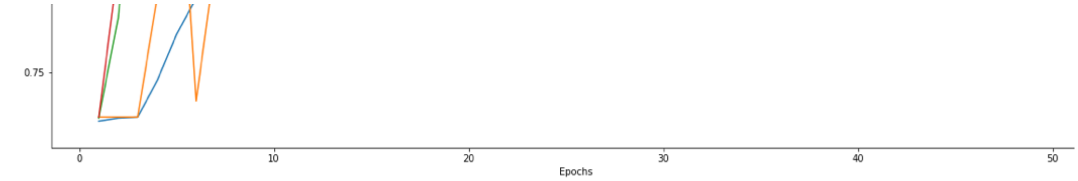

Training & Testing Accuracy Current Model vs Baseline
Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy
Epochs

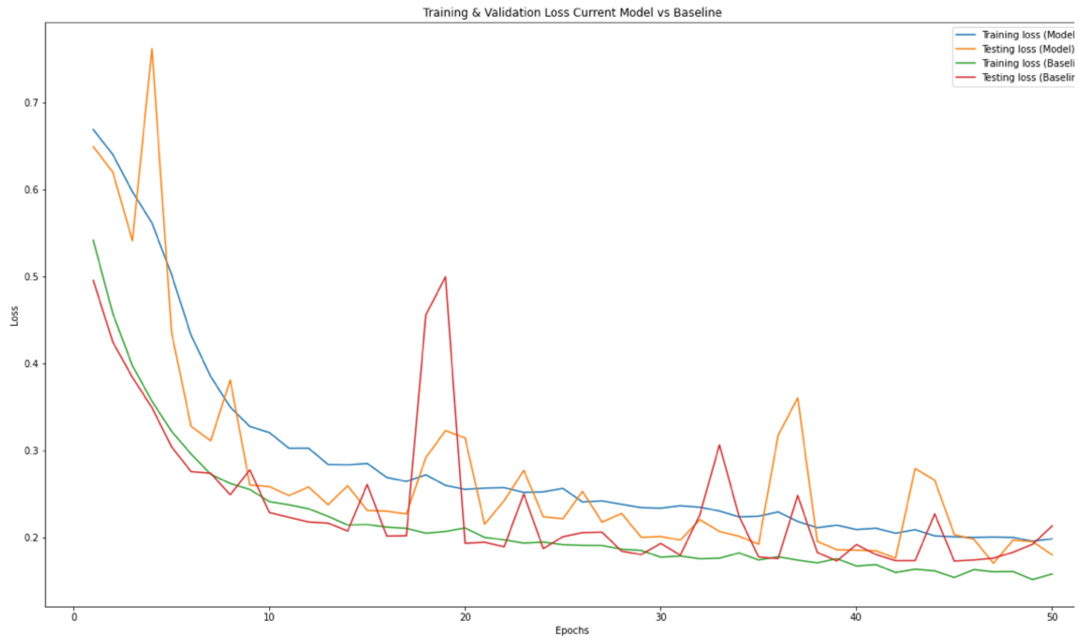

Training & Validation Loss Current Model vs Baseline
Loss
Epochs

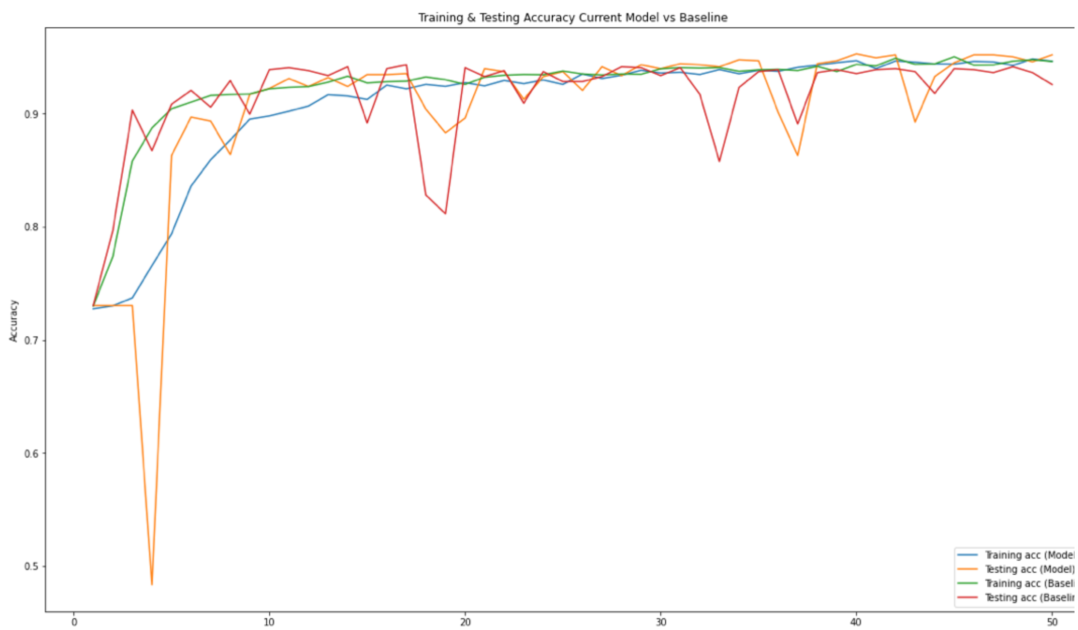

Training & Testing Accuracy Current Model vs Baseline
Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy
Epochs

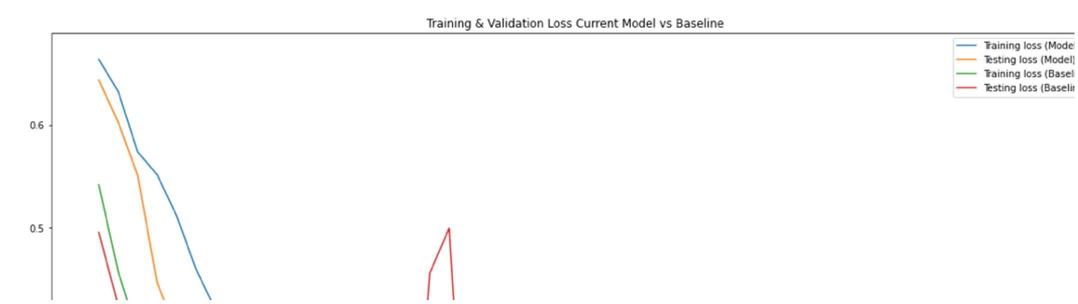

Training & Validation Loss Current Model vs Baseline
Loss
Epochs

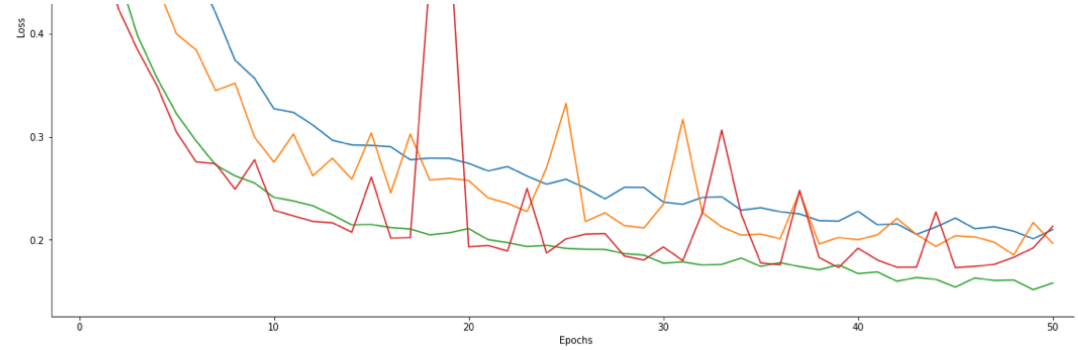

Training & Testing Accuracy Current Model vs Baseline
Training acc (Model)
Testing acc (Model)
Training acc (Baseline)
Testing acc (Baseline)
Accuracy

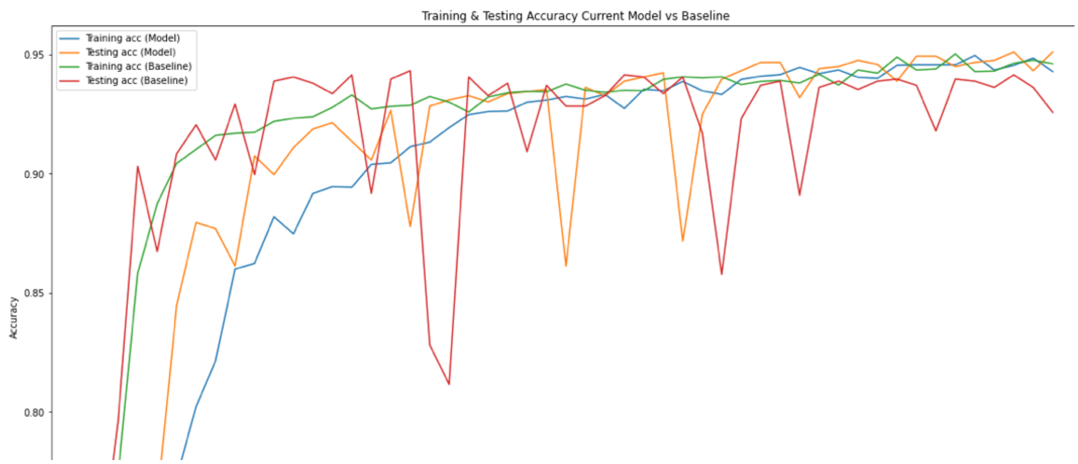

0.75
0
10
20
30
40
50
Epochs


Training & Validation Loss Current Model vs Baseline
Training loss (Model
Testing loss (Model)
Training loss (Baseli
Testing loss (Baselir
Loss
Epochs


Training & Testing Accuracy Current Model vs Baseline
Training acc (Model
Testing acc (Model)
Training acc (Baseli
Testing acc (Baselir
Accuracy
Epochs


Training & Validation Loss Current Model vs Baseline
Training loss (Model
Testing loss (Model)
Training loss (Basel
Testing loss (Baselir





In [49]:

```
# Check model run-time

end = datetime.datetime.now()
elapsed = end - start
print('Training with data augmentation took a total of {}'.format
```

```
Training with data augmentation took a total of 0:52:57.613534
```

In [50]:

```
# Print model results: Loss and Accuracy

for item in range(0,len(fold_results_2)):
    print('Fold',item+1,":\n")
    pprint.pprint(fold_results_2[item])
    print('\n')
```

```
Fold 1 :

{'Test Results:': [0.21654142855453656, 0.9441535770372986],
 'Train Results:': [0.20279987057071708, 0.9509910694837725]}


Fold 2 :

{'Test Results:': [0.22888071015867265, 0.9380453745940177],
 'Train Results:': [0.2202661924179383, 0.9326944020910477]}
```

```
Fold 3 :

{'Test Results:': [0.17284238736354868, 0.9493891791315395],
 'Train Results:': [0.15459524097330454, 0.9623175778827281]}


Fold 4 :

{'Test Results:': [0.1799898976638887, 0.9520069801786599],
 'Train Results:': [0.16929529276880348, 0.9566543236767588]}


Fold 5 :

{'Test Results:': [0.19644602792529328, 0.9511343798296198],
 'Train Results:': [0.17017031406309838, 0.9509910694837725]}
```

Looking at our results, it appears there is not much change. Let's compare all of our models and check the results.

Section 5: Results

Before selecting our best model, let's compare the results between our two models.

In [51]:

```python
# Creating a function that compares fold results between our CNN

def average_list(list):
    return sum(list)/len(list)

def fold_average(model1,model2,data, metric):
    if metric == 'Accuracy':
        if data == "Train":
            accuracy_list_1 = []
            for i in range(0,5):
                accuracy_list_1.append(list(model1[i].items())[0]
            print("Model 1 Average Accuracy:",average_list(accura
            accuracy_list_2 = []
            for i in range(0,5):
                accuracy_list_2.append(list(model2[i].items())[0]
            print("Model 2 Average Accuracy:",average_list(accura
            print('Difference between 2 Models:',abs(average_list
        if data == "Test":
            accuracy_list_1 = []
            for i in range(0,5):
                accuracy_list_1.append(list(model1[i].items())[1]
            print("Model 1 Average Accuracy:",average_list(accura
            accuracy_list_2 = []
```