Data Science - Module 4 - Final Project Submission

- Student Name: **James Toop**
- Student Pace: **Self Paced**
- Scheduled project review date/time: **Wednesday, 8th September 2021 - 9.30pm BST**
- Instructor name: **Jeff Herman**
- Blog post URL: **https://toopster.github.io/time_poor_not_poor_time_management (https://toopster.github.io/time_poor_not_poor_time_management)**

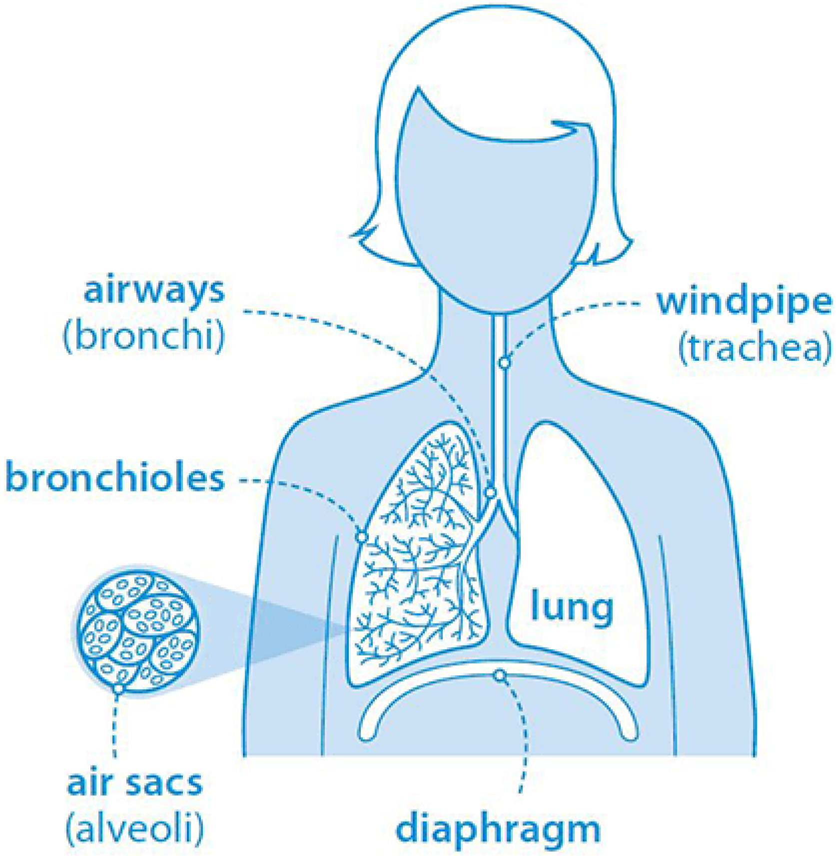
Table of Contents

---

1. Business Case and Project Purpose

1A. What is Pneumonia?

Pneumonia is an acute respiratory infection affecting the tiny air sacs in the lungs, called alveoli. When a patient has pneumonia, these air sacs get swollen (inflamed) and fill with fluid making it harder for them to breathe, even painful, and limits oxygen intake.



More people get pneumonia in winter. This is because respiratory viral infections that spread easily from person to person, such as flu, are more common in the winter, and these increase the risk of developing pneumonia. Most people with pneumonia can be completely cured, but it can be life-threatening particularly for people in "high risk" groups such as:

- babies and very young children
- elderly people
- people who smoke
- people with other health conditions, such as asthma, cystic fibrosis, or a heart, kidney or liver condition
- people with a weakened immune system – for example, as a result of a recent illness, such as flu, having HIV or AIDS, having chemotherapy, or taking medicine after an organ transplant

According to the [World Health Organisation (https://www.who.int/health-topics/pneumonia#tab=tab_1)](https://www.who.int/health-topics/pneumonia#tab=tab_1), pneumonia is the single largest infectious cause of death in children worldwide.

1B. What causes Pneumonia?

Many kinds of bacteria and viruses can cause pneumonia including coronavirus (COVID-19).

The most common type of pneumonia is **community-acquired pneumonia**, which is when pneumonia affects somebody who is not already in hospital. The most common cause of community-acquired pneumonia is a bacterium called Streptococcus pneumoniae but there are many other causes.

Other types include:

- **viral pneumonia** – caused by a virus, such as coronavirus
- **aspiration pneumonia** – caused by breathing in vomit, a foreign object, such as a peanut, or a harmful substance, such as smoke or a chemical

- **fungal pneumonia** – rare in the UK and more likely to affect people with a weakened immune system
- **hospital-acquired pneumonia** – pneumonia that develops in hospital while being treated for another condition or having an operation; people in intensive care on breathing machines are particularly at risk of developing ventilator-associated pneumonia

There are two types of vaccine available for pneumonia. They protect against the most common cause of pneumonia, the bacterium Streptococcus pneumoniae.

1C. Diagnosing Pneumonia

Community-acquired pneumonia can be difficult to diagnose because it shares many symptoms with other conditions, such as the common cold, bronchitis and asthma.

A doctor may be able to diagnose pneumonia by asking about the patient's symptoms and examining their chest but a chest X-ray is often required to confirm the presence of pneumonia.

The clinical judgement of health professionals in diagnosing pneumonia in primary care has been studied. One study demonstrated that GPs' clinical judgement had a negative predictive value (correctly ruling out pneumonia) of 96%, but a sensitivity (diagnosis after history and physical examination) of only 29%; meaning 71% of pneumonias evident on X-ray had not been suspected clinically (Van Vugt et al, 2013). The study highlighted that health professionals needed additional support to be able to consistently detect pneumonia in primary care.

1D. Project Purpose

The purpose of this project is to use Data Science and deep learning techniques to build a model that can classify whether a given patient has pneumonia, given a chest x-ray image.

Treatment of patients with bacterial pneumonia can be managed using antibiotics but the speed of intervention is important in ensuring a successful outcome. In the UK, if the patient has been admitted to hospital, treatment should be administered within 4 hours of admission.

Clearly the intention of the final model would not be to replace the expertise of the doctor but instead to augment, assist and speed up the prioritisation and treatment of patients with pneumonia.

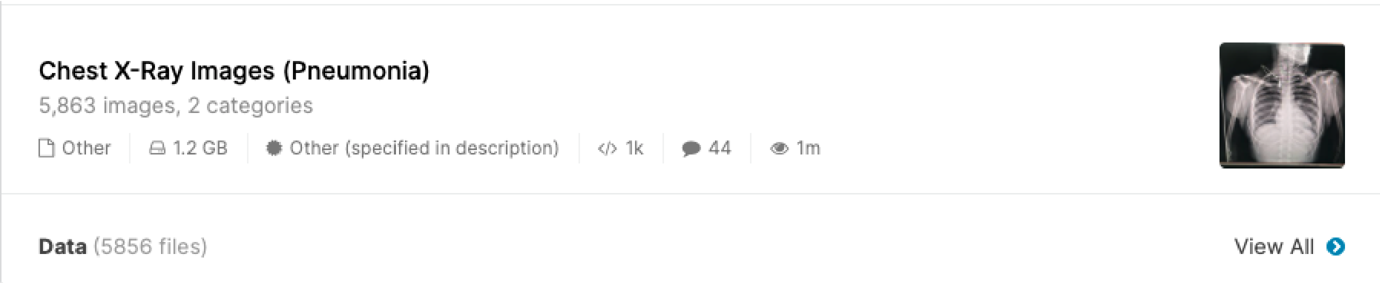
The model might also have application in locations where experienced doctors or radiological examiners are not necessarily immediately available and there may be delays in getting the x-ray analysed. This may also facilitate the treatment of patients at a community level rather than requiring longer term and expensive hospitalisation.

Sources:
* British Lung Foundation (https://www.blf.org.uk/)
* National Health Service (NHS) UK (https://www.nhs.uk/)
* World Health Organisation (WHO) (https://www.who.int/)
* Nursing Times (https://www.nursingtimes.net/)

2. Exploratory Data Analysis (EDA)

2A. The Dataset

This notebook uses x-ray images of paediatric patients to identify whether or not they have pneumonia. The dataset comes from Kermany et al. on Mendeley (https://data.mendeley.com/datasets/rscbjbr9sj/3) , but, for ease of use, we are using a version of the dataset from Kaggle (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia) which has already been organised into train, test and val subsets.



(https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia)

IMPORTANT NOTE:
The images have not been included in the GitHub repository with this notebook and will need to be downloaded and stored in the local repository for the code to run correctly.

2B. Data Discovery

This section presents an initial step to investigate, understand and document the available data fields and relationships, highlighting any potential issues / shortcomings within the datasets supplied.

In [66]:

```
# Import the relevant libraries
import os
import time
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

import scipy
from scipy import ndimage

import numpy as np
from PIL import Image

import keras
from keras import models
from keras import layers
from keras import regularizers
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array

np.random.seed(123)
```

Training Data

In [2]:

```
# Specify directory structure for images
train_folder = 'chest_xray/train/'
train_normal = 'chest_xray/train/NORMAL/'
train_pneumonia = 'chest_xray/train/PNEUMONIA/'

# Store all the relevant image names in specific objects
train_images_normal = [file for file in os.listdir(train_normal) if file.endswith('.
train_images_pneumonia = [file for file in os.listdir(train_pneumonia) if file.endsw
```

In [3]:

```
# Preview filenames for "normal" training images
train_images_normal[0:10]
```

Out[3]:

```
['NORMAL2-IM-0927-0001.jpeg',
 'NORMAL2-IM-1056-0001.jpeg',
 'IM-0427-0001.jpeg',
 'NORMAL2-IM-1260-0001.jpeg',
 'IM-0656-0001-0001.jpeg',
 'IM-0561-0001.jpeg',
 'NORMAL2-IM-1110-0001.jpeg',
 'IM-0757-0001.jpeg',
 'NORMAL2-IM-1326-0001.jpeg',
 'NORMAL2-IM-0736-0001.jpeg']
```

In [4]:

```
# Preview filenames for "pneumonia" training images
train_images_pneumonia[0:10]
```

Out[4]:

```
['person63_bacteria_306.jpeg',
 'person1438_bacteria_3721.jpeg',
 'person755_bacteria_2659.jpeg',
 'person478_virus_975.jpeg',
 'person661_bacteria_2553.jpeg',
 'person276_bacteria_1296.jpeg',
 'person1214_bacteria_3166.jpeg',
 'person1353_virus_2333.jpeg',
 'person26_bacteria_122.jpeg',
 'person124_virus_238.jpeg']
```

In [5]:

```
# Ascertain the size of the training dataset
print('Number of training chest x-ray images that are normal:', len(train_images_no
print('Number of training chest x-ray images that have pneumonia:', len(train_images
print('\nTotal training chest x-ray images:', len(train_images_normal)+len(train_ima
```

```
Number of training chest x-ray images that are normal: 1341
Number of training chest x-ray images that have pneumonia: 3875

Total training chest x-ray images: 5216
```

Test Data

In [6]:

```
# Specify directory structure for images
test_folder = 'chest_xray/test/'
test_normal = 'chest_xray/test/NORMAL/'
test_pneumonia = 'chest_xray/test/PNEUMONIA/'

# Store all the relevant image names in specific objects
test_images_normal = [file for file in os.listdir(test_normal) if file.endswith('.jp
test_images_pneumonia = [file for file in os.listdir(test_pneumonia) if file.endswit

# Ascertain the size of the test dataset
print('Number of test chest x-ray images that are normal:', len(test_images_normal)
print('Number of test chest x-ray images that have pneumonia:', len(test_images_pneu
print('\nTotal test chest x-ray images:', len(test_images_normal)+len(test_images_pr
```

```
Number of test chest x-ray images that are normal: 234
Number of test chest x-ray images that have pneumonia: 390

Total test chest x-ray images: 624
```

Validation Data

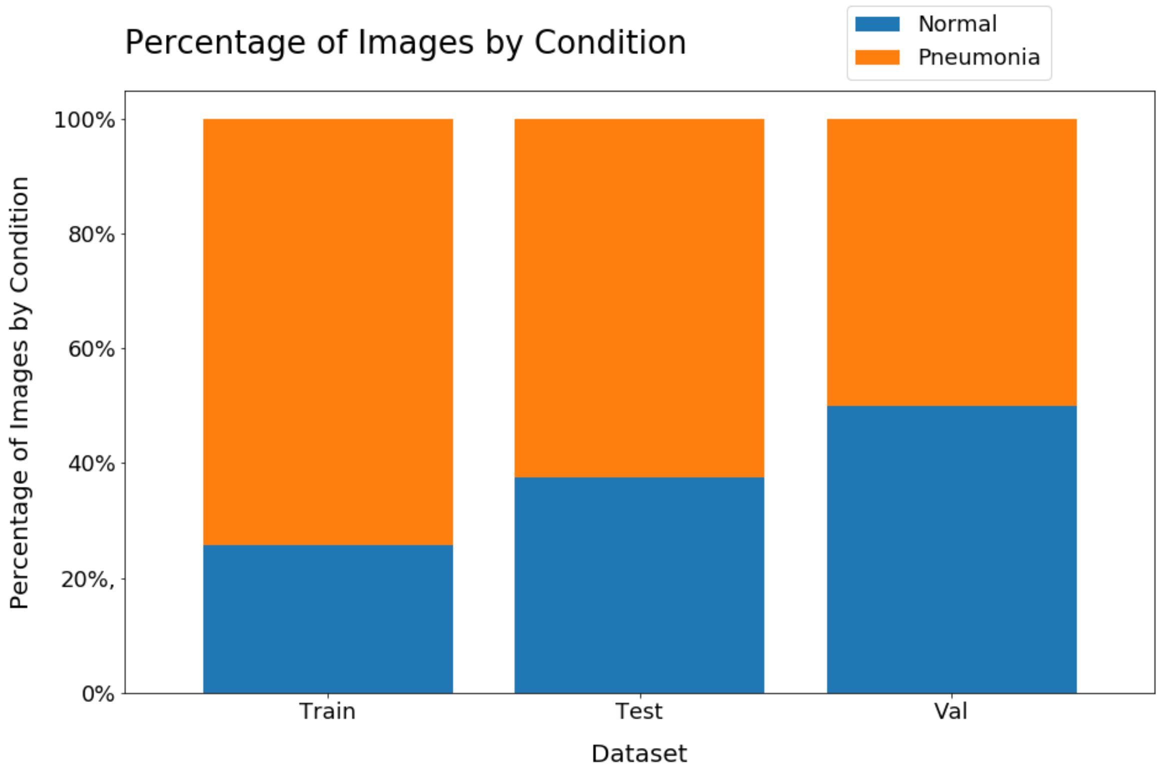
In [7]:

```
# Specify directory structure for images
val_folder = 'chest_xray/val/'
val_normal = 'chest_xray/val/NORMAL/'
val_pneumonia = 'chest_xray/val/PNEUMONIA/'

# Store all the relevant image names in specific objects
val_images_normal = [file for file in os.listdir(val_normal) if file.endswith('.jpeg
val_images_pneumonia = [file for file in os.listdir(val_pneumonia) if file.endswith(

# Ascertain the size of the validation dataset
print('Number of validation chest x-ray images that are normal:', len(val_images_no
print('Number of validation chest x-ray images that have pneumonia:', len(val_images
print('\nTotal validation chest x-ray images:', len(val_images_normal)+len(val_image
```

```
Number of validation chest x-ray images that are normal: 8
Number of validation chest x-ray images that have pneumonia: 8

Total validation chest x-ray images: 16
```

In [116]:

```
# Load the image summary data into a dataframe
image_summary = {'Dataset': ['Train','Test','Val'],
                 'Normal': [len(train_images_normal), len(test_images_normal), len(v
                 'Pneumonia': [len(train_images_pneumonia), len(test_images_pneumoni
                 'Total': [(len(train_images_normal)+len(train_images_pneumonia)),
                           (len(test_images_normal)+len(test_images_pneumonia)),
                           len(val_images_normal)+len(val_images_pneumonia)],
                }

df = pd.DataFrame(image_summary,columns=['Dataset','Normal','Pneumonia','Total'])

# Set the index of the dataframe and sort the data by the total number of images
chart_data = df.set_index('Dataset').groupby('Dataset').sum()
chart_data = chart_data.sort_values('Total', ascending=False)
chart_data = chart_data.drop('Total', axis=1)

# Create a 100% stacked bar chart to highlight the class imbalance of images by cond
chart_data_stacked = chart_data.apply(lambda x: x*100/sum(x), axis=1)

x_labels = ['Train', 'Test', 'Val']
y_labels = ['0%','20%,','40%','60%','80%','100%']

ax = chart_data_stacked.plot(kind='bar', stacked=True, figsize=(15,9), width=0.8)
ax.set_title('Percentage of Images by Condition', fontsize=26, pad=30, loc='left')
ax.set_xlabel('Dataset', fontsize=20, labelpad=16)
ax.set_ylabel('Percentage of Images by Condition', fontsize=20, labelpad=16)
ax.set_xticklabels(x_labels,
                   fontsize=18,
                   rotation=0)
ax.set_yticklabels(y_labels,
                   fontsize=18)

plt.legend(['Normal','Pneumonia'],
           bbox_to_anchor=(0.8, 1),
           loc='lower center',
           fontsize=18)
plt.show();
```



2C. Preprocessing

In [8]:

```
# Get all the data in the directory chest_xrays/train (5216 images), and reshape the
train_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
        train_folder,
        target_size = (128, 128),
        batch_size = 5216)

# Get all the data in the directory chest_xrays/test (624 images), and reshape them
test_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
        test_folder,
        target_size = (128, 128),
        batch_size = 624)

# Get all the data in the directory chest_xrays/validation (16 images), and reshape
val_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
        val_folder,
        target_size = (128, 128),
        batch_size = 16)
```

```
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

In [9]:

```
# Create the datasets
train_images, train_labels = next(train_generator)
test_images, test_labels = next(test_generator)
val_images, val_labels = next(val_generator)
```

In [10]:

```
# Explore the dataset again
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: 5216
Number of testing samples: 624
Number of validation samples: 16
train_images shape: (5216, 128, 128, 3)
train_labels shape: (5216, 2)
test_images shape: (624, 128, 128, 3)
test_labels shape: (624, 2)
val_images shape: (16, 128, 128, 3)
val_labels shape: (16, 2)
```

In [11]:

```
# Preview the training labels
train_labels[:10]
```

Out[11]:

```
array([[1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

In [12]:

```
train_img = train_images.reshape(train_images.shape[0], -1)
test_img = test_images.reshape(test_images.shape[0], -1)
val_img = val_images.reshape(val_images.shape[0], -1)

print(train_img.shape)
print(test_img.shape)
print(val_img.shape)
```

```
(5216, 49152)
(624, 49152)
(16, 49152)
```

In [13]:

```
n_features = train_img.shape[1]
n_features
```

Out[13]:

```
49152
```

In [14]:

```
train_y = np.reshape(train_labels[:,0], (5216,1))
test_y = np.reshape(test_labels[:,0], (624,1))
val_y = np.reshape(val_labels[:,0], (16,1))
```

In [15]:

```
train_y[:10]
```

Out[15]:

```
array([[1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]], dtype=float32)
```

In [16]:

```
# Function for visualising results
def visualize_results(results):
    history = results.history

    plt.figure(figsize=(20,8))
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    plt.subplot(1, 2, 1)
    plt.plot(history['val_loss'])
    plt.plot(history['loss'])
    plt.legend(['Validation Loss', 'Training Loss'], fontsize=12)
    plt.title('Loss', fontsize=18)
    plt.xlabel('Epochs', fontsize=14)
    plt.ylabel('Loss', fontsize=14)

    plt.subplot(1, 2, 2)
    plt.plot(history['val_acc'])
    plt.plot(history['acc'])
    plt.legend(['Validation Accuracy', 'Training Accuracy'], fontsize=12)
    plt.title('Accuracy', fontsize=18)
    plt.xlabel('Epochs', fontsize=14)
    plt.ylabel('Accuracy', fontsize=14)
    plt.show()
```

3. Deep Learning Neural Networks

3A. Model 1: Create a baseline network

In [17]:

```
np.random.seed(123)

# Build a baseline model
model_1 = models.Sequential()
model_1.add(layers.Dense(64, activation='tanh', input_shape=(n_features,)))
model_1.add(layers.Dense(2, activation='softmax'))

# View summary for model
model_1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 64)                3145792   
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130       
=================================================================
Total params: 3,145,922
Trainable params: 3,145,922
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
# Compile the baseline model
model_1.compile(loss='categorical_crossentropy',
                optimizer='sgd',
                metrics=['accuracy'])
```

In [19]:

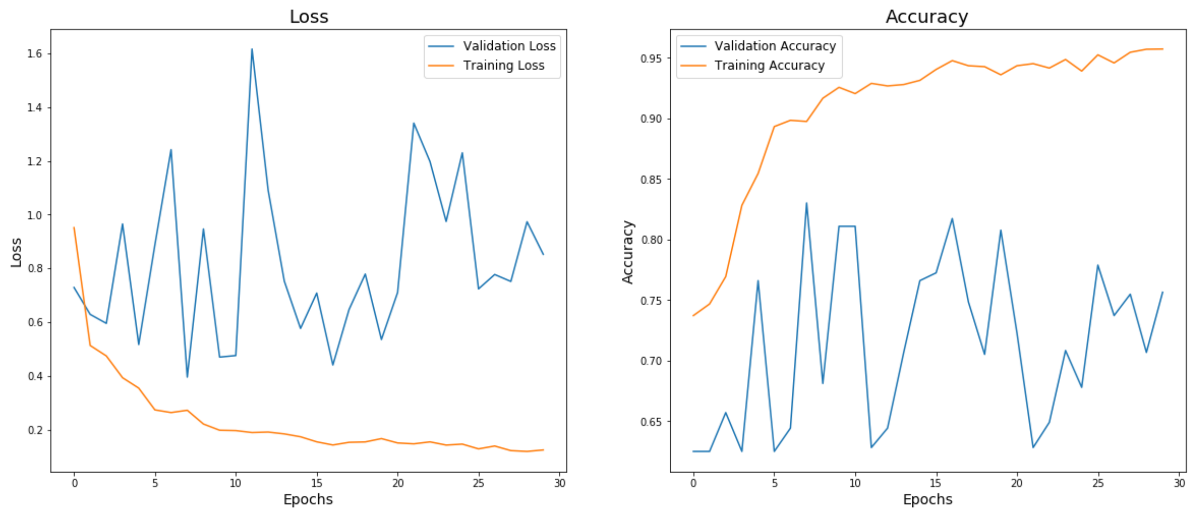
```
# Fit the baseline model
results_1 = model_1.fit(train_img,
                        train_labels,
                        epochs=30,
                        batch_size=64,
                        validation_data=(test_img, test_labels))
```

```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.951
5 - acc: 0.7372 - val_loss: 0.7289 - val_acc: 0.6250
Epoch 2/30
5216/5216 [==============================] - 5s 868us/step - loss: 0.5
130 - acc: 0.7467 - val_loss: 0.6288 - val_acc: 0.6250
Epoch 3/30
5216/5216 [==============================] - 4s 851us/step - loss: 0.4
741 - acc: 0.7694 - val_loss: 0.5953 - val_acc: 0.6571
Epoch 4/30
5216/5216 [==============================] - 4s 792us/step - loss: 0.3
931 - acc: 0.8282 - val_loss: 0.9655 - val_acc: 0.6250
Epoch 5/30
5216/5216 [==============================] - 5s 980us/step - loss: 0.3
543 - acc: 0.8545 - val_loss: 0.5165 - val_acc: 0.7660
Epoch 6/30
5216/5216 [==============================] - 5s 886us/step - loss: 0.2
733 - acc: 0.8932 - val_loss: 0.8878 - val_acc: 0.6250
Epoch 7/30
5216/5216 [==============================] - 5s 958us/step - loss: 0.2
633 - acc: 0.8984 - val_loss: 1.2420 - val_acc: 0.6442
Epoch 8/30
5216/5216 [==============================] - 4s 794us/step - loss: 0.2
719 - acc: 0.8974 - val_loss: 0.3953 - val_acc: 0.8301
Epoch 9/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.220
6 - acc: 0.9166 - val_loss: 0.9469 - val_acc: 0.6811
Epoch 10/30
5216/5216 [==============================] - 4s 856us/step - loss: 0.1
975 - acc: 0.9256 - val_loss: 0.4698 - val_acc: 0.8109
Epoch 11/30
5216/5216 [==============================] - 4s 726us/step - loss: 0.1
962 - acc: 0.9204 - val_loss: 0.4756 - val_acc: 0.8109
Epoch 12/30
5216/5216 [==============================] - 4s 758us/step - loss: 0.1
889 - acc: 0.9289 - val_loss: 1.6166 - val_acc: 0.6282
Epoch 13/30
5216/5216 [==============================] - 4s 806us/step - loss: 0.1
908 - acc: 0.9268 - val_loss: 1.0906 - val_acc: 0.6442
Epoch 14/30
5216/5216 [==============================] - 4s 689us/step - loss: 0.1
839 - acc: 0.9279 - val_loss: 0.7522 - val_acc: 0.7067
Epoch 15/30
5216/5216 [==============================] - 4s 678us/step - loss: 0.1
732 - acc: 0.9314 - val_loss: 0.5765 - val_acc: 0.7660
Epoch 16/30
5216/5216 [==============================] - 4s 687us/step - loss: 0.1
543 - acc: 0.9404 - val_loss: 0.7081 - val_acc: 0.7724
Epoch 17/30
5216/5216 [==============================] - 4s 824us/step - loss: 0.1
426 - acc: 0.9477 - val_loss: 0.4404 - val_acc: 0.8173
```

```
Epoch 18/30
5216/5216 [==============================] - 5s 996us/step - loss: 0.1
525 - acc: 0.9434 - val_loss: 0.6469 - val_acc: 0.7484
Epoch 19/30
5216/5216 [==============================] - 4s 819us/step - loss: 0.1
540 - acc: 0.9427 - val_loss: 0.7789 - val_acc: 0.7051
Epoch 20/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.166
4 - acc: 0.9360 - val_loss: 0.5352 - val_acc: 0.8077
Epoch 21/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.150
2 - acc: 0.9434 - val_loss: 0.7102 - val_acc: 0.7228
Epoch 22/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.146
9 - acc: 0.9452 - val_loss: 1.3408 - val_acc: 0.6282
Epoch 23/30
5216/5216 [==============================] - 5s 926us/step - loss: 0.1
542 - acc: 0.9415 - val_loss: 1.1977 - val_acc: 0.6490
Epoch 24/30
5216/5216 [==============================] - 4s 832us/step - loss: 0.1
423 - acc: 0.9486 - val_loss: 0.9746 - val_acc: 0.7083
Epoch 25/30
5216/5216 [==============================] - 4s 806us/step - loss: 0.1
458 - acc: 0.9390 - val_loss: 1.2302 - val_acc: 0.6779
Epoch 26/30
5216/5216 [==============================] - 4s 745us/step - loss: 0.1
281 - acc: 0.9525 - val_loss: 0.7236 - val_acc: 0.7788
Epoch 27/30
5216/5216 [==============================] - 4s 757us/step - loss: 0.1
388 - acc: 0.9457 - val_loss: 0.7772 - val_acc: 0.7372
Epoch 28/30
5216/5216 [==============================] - 4s 779us/step - loss: 0.1
217 - acc: 0.9546 - val_loss: 0.7514 - val_acc: 0.7548
Epoch 29/30
5216/5216 [==============================] - 4s 768us/step - loss: 0.1
186 - acc: 0.9571 - val_loss: 0.9737 - val_acc: 0.7067
Epoch 30/30
5216/5216 [==============================] - 4s 786us/step - loss: 0.1
241 - acc: 0.9572 - val_loss: 0.8525 - val_acc: 0.7564
```

In [20]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_1)
```



In [21]:

```
# Evaluate the training results
results_1_train = model_1.evaluate(train_img, train_labels)
results_1_train
```

```
5216/5216 [==============================] - 1s 278us/step
```

Out[21]:

```
[0.09718865230450967, 0.964532208588957]
```

In [22]:

```
# Evaluate the test results
results_1_test = model_1.evaluate(test_img, test_labels)
results_1_test
```

```
624/624 [==============================] - 0s 302us/step
```

Out[22]:

```
[0.8524664854392027, 0.7564102564102564]
```

Conclusion

It's starting point and the training accuracy is almost 97%, but it is not entirely clear as to whether the validation accuracy is converging or not. In addition there is a lot of fluctuation in the validation accuracy ranging between a little over 50% (so no better than a coin toss) to almost 50%.

The model is clearly overfitting the training data and is not generalising well when shown unseen data.

3B. Model 2: Deepen the network and increase the number of neurons in each layer

In [23]:

```
np.random.seed(123)

# Build a deeper model
model_2 = models.Sequential()
model_2.add(layers.Dense(300, activation='tanh', input_shape=(n_features,)))
model_2.add(layers.Dense(100, activation='tanh'))
model_2.add(layers.Dense(2, activation='softmax'))

# View summary for model
model_2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 300)               14745900
_________________________________________________________________
dense_4 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 202
=================================================================
Total params: 14,776,202
Trainable params: 14,776,202
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

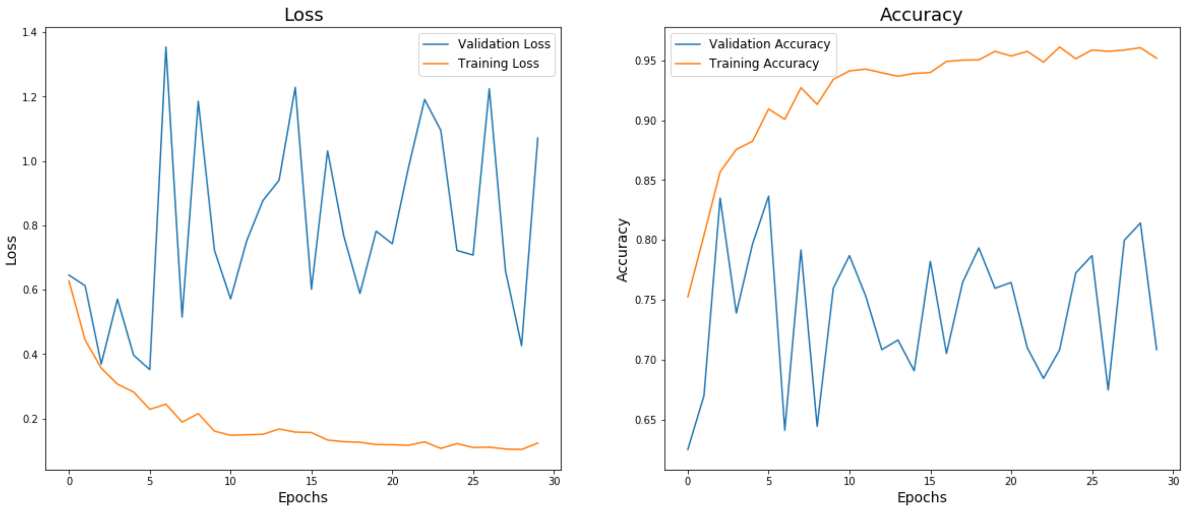
```
# Compile the deeper model
model_2.compile(loss='categorical_crossentropy',
                optimizer='sgd',
                metrics=['accuracy'])

# Fit the deeper model
results_2 = model_2.fit(train_img,
                        train_labels,
                        batch_size=64,
                        epochs=30,
                        validation_data=(test_img, test_labels))
```

```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 13s 3ms/step - loss: 0.6
268 - acc: 0.7523 - val_loss: 0.6454 - val_acc: 0.6250
Epoch 2/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.4
440 - acc: 0.8039 - val_loss: 0.6126 - val_acc: 0.6699
Epoch 3/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.3
567 - acc: 0.8568 - val_loss: 0.3697 - val_acc: 0.8349
Epoch 4/30
5216/5216 [==============================] - 15s 3ms/step - loss: 0.3
069 - acc: 0.8756 - val_loss: 0.5706 - val_acc: 0.7388
Epoch 5/30
5216/5216 [==============================] - 14s 3ms/step - loss: 0.2
827 - acc: 0.8823 - val_loss: 0.3968 - val_acc: 0.7965
Epoch 6/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.2
288 - acc: 0.9093 - val_loss: 0.3520 - val_acc: 0.8365
Epoch 7/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.2
447 - acc: 0.9007 - val_loss: 1.3535 - val_acc: 0.6410
Epoch 8/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
891 - acc: 0.9271 - val_loss: 0.5153 - val_acc: 0.7917
Epoch 9/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.2
155 - acc: 0.9132 - val_loss: 1.1850 - val_acc: 0.6442
Epoch 10/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.1
611 - acc: 0.9340 - val_loss: 0.7227 - val_acc: 0.7596
Epoch 11/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
482 - acc: 0.9411 - val_loss: 0.5716 - val_acc: 0.7869
Epoch 12/30
5216/5216 [==============================] - 14s 3ms/step - loss: 0.1
498 - acc: 0.9427 - val_loss: 0.7528 - val_acc: 0.7532
Epoch 13/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
512 - acc: 0.9396 - val_loss: 0.8770 - val_acc: 0.7083
Epoch 14/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
678 - acc: 0.9367 - val_loss: 0.9400 - val_acc: 0.7163
Epoch 15/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
581 - acc: 0.9390 - val_loss: 1.2281 - val_acc: 0.6907
Epoch 16/30
```

```
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
568 - acc: 0.9398 - val_loss: 0.6014 - val_acc: 0.7821
Epoch 17/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
334 - acc: 0.9490 - val_loss: 1.0311 - val_acc: 0.7051
Epoch 18/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
283 - acc: 0.9502 - val_loss: 0.7676 - val_acc: 0.7644
Epoch 19/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
265 - acc: 0.9503 - val_loss: 0.5886 - val_acc: 0.7933
Epoch 20/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
194 - acc: 0.9574 - val_loss: 0.7818 - val_acc: 0.7596
Epoch 21/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
189 - acc: 0.9536 - val_loss: 0.7428 - val_acc: 0.7644
Epoch 22/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
172 - acc: 0.9574 - val_loss: 0.9787 - val_acc: 0.7099
Epoch 23/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
278 - acc: 0.9484 - val_loss: 1.1907 - val_acc: 0.6843
Epoch 24/30
5216/5216 [==============================] - 13s 2ms/step - loss: 0.1
075 - acc: 0.9611 - val_loss: 1.0949 - val_acc: 0.7083
Epoch 25/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.1
223 - acc: 0.9513 - val_loss: 0.7220 - val_acc: 0.7724
Epoch 26/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
104 - acc: 0.9586 - val_loss: 0.7076 - val_acc: 0.7869
Epoch 27/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
113 - acc: 0.9574 - val_loss: 1.2240 - val_acc: 0.6747
Epoch 28/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.1
056 - acc: 0.9586 - val_loss: 0.6594 - val_acc: 0.7997
Epoch 29/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
043 - acc: 0.9605 - val_loss: 0.4263 - val_acc: 0.8141
Epoch 30/30
5216/5216 [==============================] - 12s 2ms/step - loss: 0.1
240 - acc: 0.9517 - val_loss: 1.0710 - val_acc: 0.7083
```

In [25]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_2)
```



In [26]:

```
# Evaluate the training results
results_2_train = model_2.evaluate(train_img, train_labels)
results_2_train
```

```
5216/5216 [==============================] - 3s 668us/step
```

Out[26]:

```
[0.11773227483171261, 0.9514953987730062]
```

In [27]:

```
# Evaluate the test results
results_2_test = model_2.evaluate(test_img, test_labels)
results_2_test
```

```
624/624 [==============================] - 0s 707us/step
```

Out[27]:

```
[1.0709681572058263, 0.7083333333333334]
```

Conclusion

Deepening the neural network seems does not appear to have improved the model and, if anything, the accuracy against both the training and validation sets has reduced.

3C. Model 3: A deeper network but with a different activation type and reduce the number of neurons

In [28]:

```
np.random.seed(123)

# Build a deeper model with less neurons and change activation type
model_3 = models.Sequential()
model_3.add(layers.Dense(64, activation='relu', input_shape=(n_features,)))
model_3.add(layers.Dense(32, activation='relu'))
model_3.add(layers.Dense(16, activation='relu'))
model_3.add(layers.Dense(2, activation='softmax'))

model_3.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_6 (Dense)              (None, 64)                3145792   
_________________________________________________________________
dense_7 (Dense)              (None, 32)                2080      
_________________________________________________________________
dense_8 (Dense)              (None, 16)                528       
_________________________________________________________________
dense_9 (Dense)              (None, 2)                 34        
=================================================================
Total params: 3,148,434
Trainable params: 3,148,434
Non-trainable params: 0
_________________________________________________________________
```

In [29]:

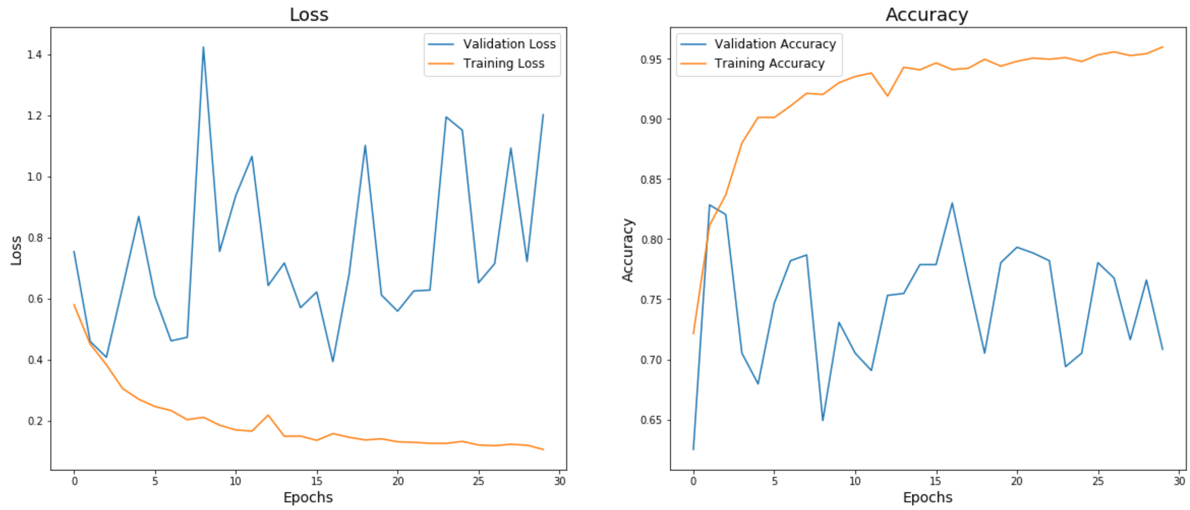
```
# Compile model
model_3.compile(loss='categorical_crossentropy',
                optimizer='sgd',
                metrics=['accuracy'])

# Fit model
results_3 = model_3.fit(train_img,
                        train_labels,
                        batch_size=64,
                        epochs=30,
                        validation_data=(test_img, test_labels))
```

```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 5s 866us/step - loss: 0.
5801 - acc: 0.7214 - val_loss: 0.7539 - val_acc: 0.6250
Epoch 2/30
5216/5216 [==============================] - 5s 867us/step - loss: 0.
4513 - acc: 0.8115 - val_loss: 0.4598 - val_acc: 0.8285
Epoch 3/30
5216/5216 [==============================] - 4s 749us/step - loss: 0.
3838 - acc: 0.8368 - val_loss: 0.4081 - val_acc: 0.8205
Epoch 4/30
5216/5216 [==============================] - 5s 919us/step - loss: 0.
3059 - acc: 0.8800 - val_loss: 0.6361 - val_acc: 0.7051
Epoch 5/30
5216/5216 [==============================] - 4s 796us/step - loss: 0.
2707 - acc: 0.9015 - val_loss: 0.8694 - val_acc: 0.6795
Epoch 6/30
5216/5216 [==============================] - 7s 1ms/step - loss: 0.24
68 - acc: 0.9013 - val_loss: 0.6069 - val_acc: 0.7468
Epoch 7/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.23
37 - acc: 0.9109 - val_loss: 0.4619 - val_acc: 0.7821
Epoch 8/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.20
36 - acc: 0.9214 - val_loss: 0.4735 - val_acc: 0.7869
Epoch 9/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.21
16 - acc: 0.9204 - val_loss: 1.4242 - val_acc: 0.6490
Epoch 10/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.18
56 - acc: 0.9302 - val_loss: 0.7547 - val_acc: 0.7308
Epoch 11/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.17
01 - acc: 0.9354 - val_loss: 0.9379 - val_acc: 0.7051
Epoch 12/30
5216/5216 [==============================] - 7s 1ms/step - loss: 0.16
63 - acc: 0.9383 - val_loss: 1.0660 - val_acc: 0.6907
Epoch 13/30
5216/5216 [==============================] - 7s 1ms/step - loss: 0.21
86 - acc: 0.9191 - val_loss: 0.6432 - val_acc: 0.7532
Epoch 14/30
5216/5216 [==============================] - 4s 830us/step - loss: 0.
1494 - acc: 0.9431 - val_loss: 0.7168 - val_acc: 0.7548
Epoch 15/30
5216/5216 [==============================] - 4s 814us/step - loss: 0.
1501 - acc: 0.9410 - val_loss: 0.5706 - val_acc: 0.7788
Epoch 16/30
```

```
5216/5216 [==============================] - 4s 789us/step - loss: 0.
1360 - acc: 0.9467 - val_loss: 0.6219 - val_acc: 0.7788
Epoch 17/30
5216/5216 [==============================] - 4s 746us/step - loss: 0.
1580 - acc: 0.9411 - val_loss: 0.3938 - val_acc: 0.8301
Epoch 18/30
5216/5216 [==============================] - 4s 720us/step - loss: 0.
1461 - acc: 0.9423 - val_loss: 0.6789 - val_acc: 0.7660
Epoch 19/30
5216/5216 [==============================] - 4s 762us/step - loss: 0.
1374 - acc: 0.9498 - val_loss: 1.1023 - val_acc: 0.7051
Epoch 20/30
5216/5216 [==============================] - 4s 715us/step - loss: 0.
1410 - acc: 0.9440 - val_loss: 0.6120 - val_acc: 0.7804
Epoch 21/30
5216/5216 [==============================] - 4s 729us/step - loss: 0.
1315 - acc: 0.9480 - val_loss: 0.5590 - val_acc: 0.7933
Epoch 22/30
5216/5216 [==============================] - 4s 709us/step - loss: 0.
1295 - acc: 0.9507 - val_loss: 0.6252 - val_acc: 0.7885
Epoch 23/30
5216/5216 [==============================] - 4s 688us/step - loss: 0.
1266 - acc: 0.9498 - val_loss: 0.6282 - val_acc: 0.7821
Epoch 24/30
5216/5216 [==============================] - 4s 732us/step - loss: 0.
1261 - acc: 0.9511 - val_loss: 1.1953 - val_acc: 0.6939
Epoch 25/30
5216/5216 [==============================] - 4s 714us/step - loss: 0.
1325 - acc: 0.9479 - val_loss: 1.1517 - val_acc: 0.7051
Epoch 26/30
5216/5216 [==============================] - 4s 685us/step - loss: 0.
1209 - acc: 0.9534 - val_loss: 0.6522 - val_acc: 0.7804
Epoch 27/30
5216/5216 [==============================] - 4s 761us/step - loss: 0.
1186 - acc: 0.9559 - val_loss: 0.7150 - val_acc: 0.7676
Epoch 28/30
5216/5216 [==============================] - 5s 893us/step - loss: 0.
1232 - acc: 0.9528 - val_loss: 1.0933 - val_acc: 0.7163
Epoch 29/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.12
02 - acc: 0.9544 - val_loss: 0.7218 - val_acc: 0.7660
Epoch 30/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.10
63 - acc: 0.9599 - val_loss: 1.2025 - val_acc: 0.7083
```

In [30]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_3)
```



In [31]:

```
# Evaluate the training results
results_3_train = model_3.evaluate(train_img, train_labels)
results_3_train
```

```
5216/5216 [==============================] - 2s 428us/step
```

Out[31]:

```
[0.12857253192438303, 0.9526457055214724]
```

In [32]:

```
# Evaluate the test results
results_3_test = model_3.evaluate(test_img, test_labels)
results_3_test
```

```
624/624 [==============================] - 0s 344us/step
```

Out[32]:

```
[1.20247494104581, 0.7083333333333334]
```

Conclusion

A deeper network does not appear to resolve the issue with poor generalisation and, in conjunction with this, changing the activation type and reducing the number of neurons also does not appear to offer any improvement on the model.

Reading up on the issue of generalisation, reducing the learning rate and introducing regularization could be beneficial.

3D. Model 4: Adding some regularization and reducing the learning rate

In [33]:

```
np.random.seed(123)

# Build the model
model_4 = models.Sequential()
model_4.add(layers.Dense(64, activation='relu', input_shape=(n_features,)))
model_4.add(layers.Dense(32, kernel_regularizer=regularizers.l2(0.005), activation=
model_4.add(layers.Dense(2, activation='softmax'))

# View summary for model
model_4.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_10 (Dense)             (None, 64)                3145792   
_________________________________________________________________
dense_11 (Dense)             (None, 32)                2080      
_________________________________________________________________
dense_12 (Dense)             (None, 2)                 66        
=================================================================
Total params: 3,147,938
Trainable params: 3,147,938
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

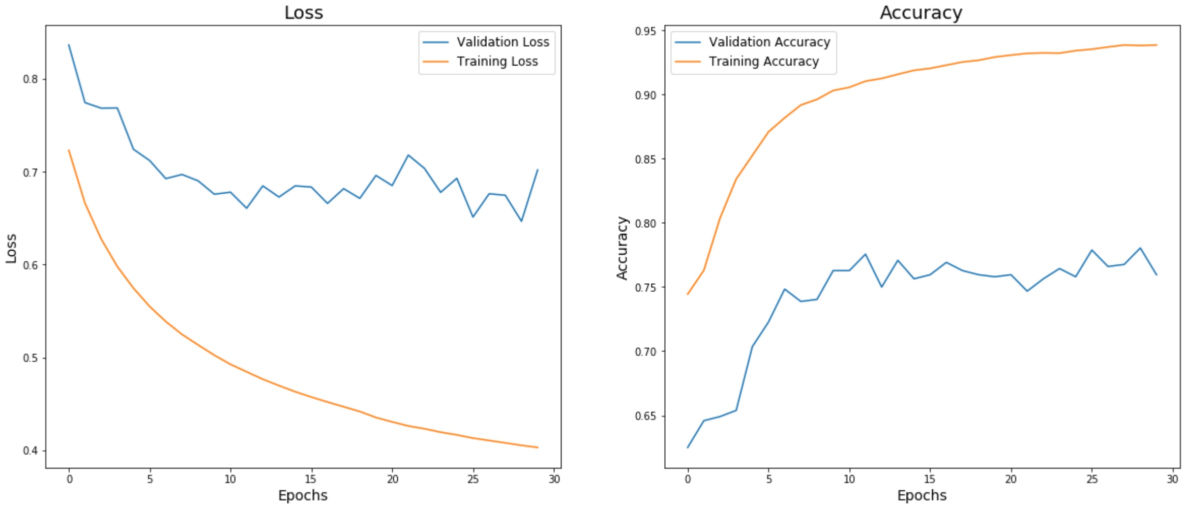
```
optimizer = keras.optimizers.SGD(0.0001)

# Compile model
model_4.compile(loss='categorical_crossentropy',
                optimizer=optimizer,
                metrics=['accuracy'])

# Fit model
results_4 = model_4.fit(train_img,
                        train_labels,
                        batch_size=64,
                        epochs=30,
                        validation_data=(test_img, test_labels))
```

```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 5s 886us/step - loss: 0.7
232 - acc: 0.7444 - val_loss: 0.8365 - val_acc: 0.6250
Epoch 2/30
5216/5216 [==============================] - 4s 752us/step - loss: 0.6
662 - acc: 0.7630 - val_loss: 0.7745 - val_acc: 0.6458
Epoch 3/30
5216/5216 [==============================] - 4s 803us/step - loss: 0.6
277 - acc: 0.8037 - val_loss: 0.7685 - val_acc: 0.6490
Epoch 4/30
5216/5216 [==============================] - 4s 711us/step - loss: 0.5
980 - acc: 0.8342 - val_loss: 0.7687 - val_acc: 0.6538
Epoch 5/30
5216/5216 [==============================] - 4s 780us/step - loss: 0.5
747 - acc: 0.8526 - val_loss: 0.7242 - val_acc: 0.7035
Epoch 6/30
5216/5216 [==============================] - 5s 868us/step - loss: 0.5
550 - acc: 0.8710 - val_loss: 0.7121 - val_acc: 0.7228
Epoch 7/30
5216/5216 [==============================] - 4s 792us/step - loss: 0.5
388 - acc: 0.8819 - val_loss: 0.6926 - val_acc: 0.7484
Epoch 8/30
5216/5216 [==============================] - 4s 803us/step - loss: 0.5
249 - acc: 0.8919 - val_loss: 0.6971 - val_acc: 0.7388
Epoch 9/30
5216/5216 [==============================] - 4s 824us/step - loss: 0.5
135 - acc: 0.8963 - val_loss: 0.6902 - val_acc: 0.7404
Epoch 10/30
5216/5216 [==============================] - 5s 878us/step - loss: 0.5
025 - acc: 0.9032 - val_loss: 0.6758 - val_acc: 0.7628
Epoch 11/30
5216/5216 [==============================] - 4s 855us/step - loss: 0.4
927 - acc: 0.9057 - val_loss: 0.6780 - val_acc: 0.7628
Epoch 12/30
5216/5216 [==============================] - 5s 871us/step - loss: 0.4
846 - acc: 0.9105 - val_loss: 0.6607 - val_acc: 0.7756
Epoch 13/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.476
7 - acc: 0.9126 - val_loss: 0.6848 - val_acc: 0.7500
Epoch 14/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.469
7 - acc: 0.9158 - val_loss: 0.6728 - val_acc: 0.7708
Epoch 15/30
5216/5216 [==============================] - 6s 1ms/step - loss: 0.463
```

```
2 - acc: 0.9189 - val_loss: 0.6848 - val_acc: 0.7564
Epoch 16/30
5216/5216 [==============================] - 5s 923us/step - loss: 0.4
574 - acc: 0.9204 - val_loss: 0.6836 - val_acc: 0.7596
Epoch 17/30
5216/5216 [==============================] - 5s 885us/step - loss: 0.4
521 - acc: 0.9229 - val_loss: 0.6659 - val_acc: 0.7692
Epoch 18/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.447
0 - acc: 0.9254 - val_loss: 0.6818 - val_acc: 0.7628
Epoch 19/30
5216/5216 [==============================] - 5s 908us/step - loss: 0.4
419 - acc: 0.9268 - val_loss: 0.6714 - val_acc: 0.7596
Epoch 20/30
5216/5216 [==============================] - 4s 786us/step - loss: 0.4
354 - acc: 0.9293 - val_loss: 0.6961 - val_acc: 0.7580
Epoch 21/30
5216/5216 [==============================] - 5s 898us/step - loss: 0.4
307 - acc: 0.9308 - val_loss: 0.6851 - val_acc: 0.7596
Epoch 22/30
5216/5216 [==============================] - 5s 1ms/step - loss: 0.426
3 - acc: 0.9321 - val_loss: 0.7180 - val_acc: 0.7468
Epoch 23/30
5216/5216 [==============================] - 4s 751us/step - loss: 0.4
233 - acc: 0.9325 - val_loss: 0.7036 - val_acc: 0.7564
Epoch 24/30
5216/5216 [==============================] - 4s 716us/step - loss: 0.4
196 - acc: 0.9323 - val_loss: 0.6778 - val_acc: 0.7644
Epoch 25/30
5216/5216 [==============================] - 4s 793us/step - loss: 0.4
167 - acc: 0.9342 - val_loss: 0.6930 - val_acc: 0.7580
Epoch 26/30
5216/5216 [==============================] - 4s 711us/step - loss: 0.4
132 - acc: 0.9354 - val_loss: 0.6512 - val_acc: 0.7788
Epoch 27/30
5216/5216 [==============================] - 4s 838us/step - loss: 0.4
107 - acc: 0.9371 - val_loss: 0.6763 - val_acc: 0.7660
Epoch 28/30
5216/5216 [==============================] - 4s 798us/step - loss: 0.4
081 - acc: 0.9387 - val_loss: 0.6747 - val_acc: 0.7676
Epoch 29/30
5216/5216 [==============================] - 4s 781us/step - loss: 0.4
055 - acc: 0.9383 - val_loss: 0.6467 - val_acc: 0.7804
Epoch 30/30
5216/5216 [==============================] - 4s 737us/step - loss: 0.4
032 - acc: 0.9387 - val_loss: 0.7020 - val_acc: 0.7596
```

In [35]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_4)
```



In [36]:

```
# Evaluate the training results
results_4_train = model_4.evaluate(train_img, train_labels)
results_4_train
```

```
5216/5216 [==============================] - 1s 275us/step
```

Out[36]:

```
[0.4013467974092331, 0.9369248466257669]
```

In [37]:

```
# Evaluate the training results
results_4_test = model_4.evaluate(test_img, test_labels)
results_4_test
```

```
624/624 [==============================] - 0s 333us/step
```

Out[37]:

```
[0.7019937435785929, 0.7596153846153846]
```

Conclusion

Reducing the learning rate has indeed improved the model with the validation loss clearly converging and the wild fluctuations smoothing the curve.

There is, however, still quite a difference between the training accuracy and the validation accuracy which suggests the model is overfitting and still not generalising well.

3E. Model 5: Adding a dropout layer and trying other optimizers with a reduced learning rate

In [38]:

```
np.random.seed(123)

# Build the model
model_5 = models.Sequential()
model_5.add(layers.Dropout(0.3, input_shape=(n_features,)))
model_5.add(layers.Dense(64, activation='relu'))
model_5.add(layers.Dropout(0.3))
model_5.add(layers.Dense(32, kernel_regularizer=regularizers.l2(0.005), activation=
model_5.add(layers.Dropout(0.3))
model_5.add(layers.Dense(2, activation='softmax'))

# View summary for model
model_5.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dropout_1 (Dropout)          (None, 49152)             0         
_________________________________________________________________
dense_13 (Dense)             (None, 64)                3145792   
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_14 (Dense)             (None, 32)                2080      
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_15 (Dense)             (None, 2)                 66        
=================================================================
Total params: 3,147,938
Trainable params: 3,147,938
Non-trainable params: 0
_________________________________________________________________
```

In [39]:

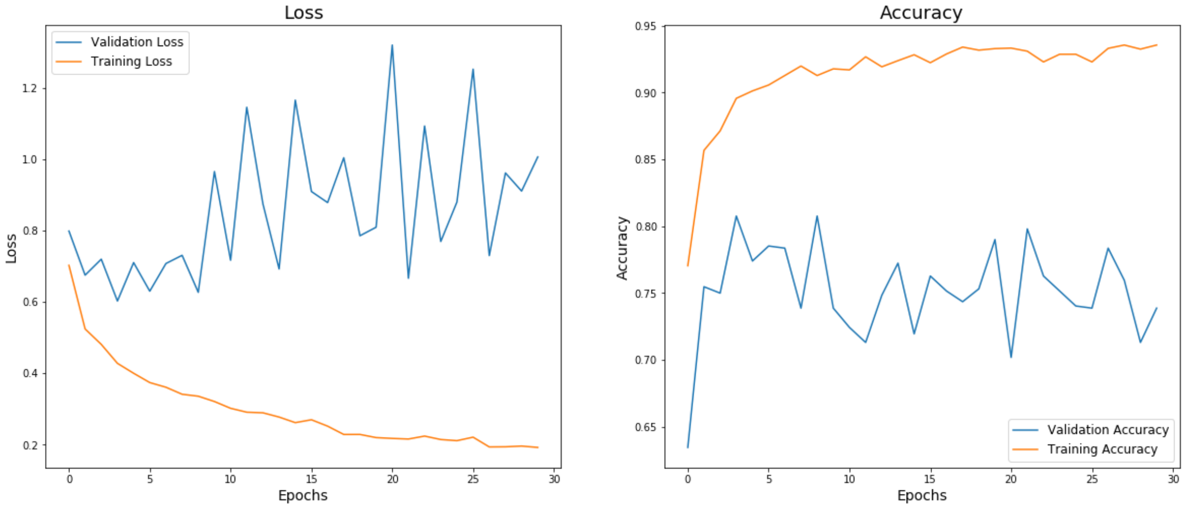
```
optimizer = keras.optimizers.Adam(0.0001)

# Compile model
model_5.compile(loss='categorical_crossentropy',
                optimizer=optimizer,
                metrics=['accuracy'])

# Fit model
results_5 = model_5.fit(train_img,
                        train_labels,
                        batch_size=64,
                        epochs=30,
                        validation_data=(test_img, test_labels))
```

```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.702
9 - acc: 0.7705 - val_loss: 0.7992 - val_acc: 0.6346
Epoch 2/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.524
5 - acc: 0.8568 - val_loss: 0.6754 - val_acc: 0.7548
Epoch 3/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.481
1 - acc: 0.8714 - val_loss: 0.7203 - val_acc: 0.7500
Epoch 4/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.428
0 - acc: 0.8957 - val_loss: 0.6028 - val_acc: 0.8077
Epoch 5/30
5216/5216 [==============================] - 10s 2ms/step - loss: 0.40
01 - acc: 0.9013 - val_loss: 0.7107 - val_acc: 0.7740
Epoch 6/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.374
0 - acc: 0.9057 - val_loss: 0.6304 - val_acc: 0.7853
Epoch 7/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.360
9 - acc: 0.9128 - val_loss: 0.7081 - val_acc: 0.7837
Epoch 8/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.341
2 - acc: 0.9199 - val_loss: 0.7311 - val_acc: 0.7388
Epoch 9/30
5216/5216 [==============================] - 8s 1ms/step - loss: 0.335
7 - acc: 0.9128 - val_loss: 0.6272 - val_acc: 0.8077
Epoch 10/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.32
08 - acc: 0.9178 - val_loss: 0.9661 - val_acc: 0.7388
Epoch 11/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.301
8 - acc: 0.9170 - val_loss: 0.7173 - val_acc: 0.7244
Epoch 12/30
5216/5216 [==============================] - 13s 2ms/step - loss: 0.29
08 - acc: 0.9268 - val_loss: 1.1465 - val_acc: 0.7131
Epoch 13/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.28
91 - acc: 0.9193 - val_loss: 0.8753 - val_acc: 0.7484
Epoch 14/30
5216/5216 [==============================] - 11s 2ms/step - loss: 0.27
72 - acc: 0.9239 - val_loss: 0.6928 - val_acc: 0.7724
Epoch 15/30
```

```
5216/5216 [==============================] - 10s 2ms/step - loss: 0.26
15 - acc: 0.9283 - val_loss: 1.1666 - val_acc: 0.7196
Epoch 16/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.269
6 - acc: 0.9224 - val_loss: 0.9101 - val_acc: 0.7628
Epoch 17/30
5216/5216 [==============================] - 10s 2ms/step - loss: 0.25
17 - acc: 0.9289 - val_loss: 0.8791 - val_acc: 0.7516
Epoch 18/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.228
4 - acc: 0.9340 - val_loss: 1.0049 - val_acc: 0.7436
Epoch 19/30
5216/5216 [==============================] - 8s 2ms/step - loss: 0.228
4 - acc: 0.9317 - val_loss: 0.7860 - val_acc: 0.7532
Epoch 20/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.219
6 - acc: 0.9329 - val_loss: 0.8099 - val_acc: 0.7901
Epoch 21/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.217
3 - acc: 0.9333 - val_loss: 1.3212 - val_acc: 0.7019
Epoch 22/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.215
6 - acc: 0.9310 - val_loss: 0.6667 - val_acc: 0.7981
Epoch 23/30
5216/5216 [==============================] - 10s 2ms/step - loss: 0.22
38 - acc: 0.9229 - val_loss: 1.0940 - val_acc: 0.7628
Epoch 24/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.214
2 - acc: 0.9287 - val_loss: 0.7699 - val_acc: 0.7516
Epoch 25/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.210
9 - acc: 0.9287 - val_loss: 0.8806 - val_acc: 0.7404
Epoch 26/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.220
7 - acc: 0.9229 - val_loss: 1.2534 - val_acc: 0.7388
Epoch 27/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.193
4 - acc: 0.9331 - val_loss: 0.7303 - val_acc: 0.7837
Epoch 28/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.193
8 - acc: 0.9356 - val_loss: 0.9622 - val_acc: 0.7596
Epoch 29/30
5216/5216 [==============================] - 9s 2ms/step - loss: 0.195
8 - acc: 0.9325 - val_loss: 0.9114 - val_acc: 0.7131
Epoch 30/30
5216/5216 [==============================] - 10s 2ms/step - loss: 0.19
20 - acc: 0.9356 - val_loss: 1.0072 - val_acc: 0.7388
```

In [40]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_5)
```



In [41]:

```
# Evaluate the training results
results_5_train = model_5.evaluate(train_img, train_labels)
results_5_train
```

```
5216/5216 [==============================] - 2s 411us/step
```

Out[41]:

```
[0.1313075432839569, 0.9631901840490797]
```

In [42]:

```
# Evaluate the training results
results_5_test = model_5.evaluate(test_img, test_labels)
results_5_test
```

```
624/624 [==============================] - 0s 526us/step
```

Out[42]:

```
[1.0072329518122551, 0.7387820512820513]
```

Conclusion

Using a different optimizer (Adam instead of SGD) was a retrograde step and again, the validation loss does not appear to be converging to a minimum.

In the next model, we will try using a Convolutional Neural Network for our model.

3F. Model 6: Building a CNN model

In [43]:

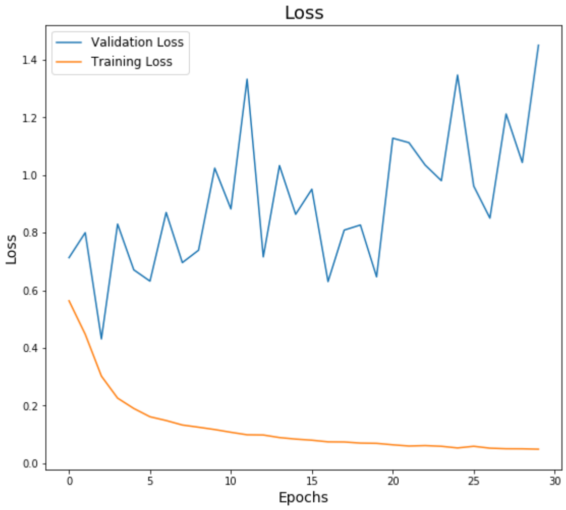
```
np.random.seed(123)

# Build the model
model_6 = models.Sequential()
model_6.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model_6.add(layers.MaxPooling2D((2, 2)))

model_6.add(layers.Conv2D(32, (4, 4), activation='relu'))
model_6.add(layers.MaxPooling2D((2, 2)))

model_6.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_6.add(layers.MaxPooling2D((2, 2)))

model_6.add(layers.Flatten())
model_6.add(layers.Dense(64, activation='relu'))
model_6.add(layers.Dense(1, activation='sigmoid'))

#Compile the model
model_6.compile(loss='binary_crossentropy',
                optimizer='sgd',
                metrics=['accuracy'])

results_6 = model_6.fit(train_images,
                        train_y,
                        epochs=30,
                        batch_size=32,
                        validation_data=(test_images, test_y))
```

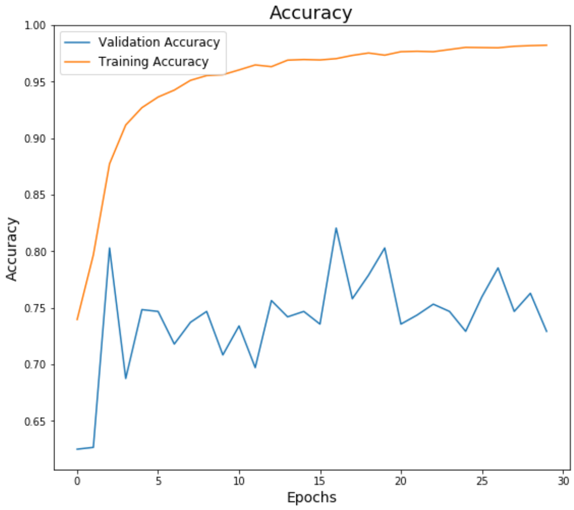
```
Train on 5216 samples, validate on 624 samples
Epoch 1/30
5216/5216 [==============================] - 166s 32ms/step - loss: 0.
5640 - acc: 0.7396 - val_loss: 0.7134 - val_acc: 0.6250
Epoch 2/30
5216/5216 [==============================] - 212s 41ms/step - loss: 0.
4479 - acc: 0.7968 - val_loss: 0.8002 - val_acc: 0.6266
Epoch 3/30
5216/5216 [==============================] - 163s 31ms/step - loss: 0.
3027 - acc: 0.8773 - val_loss: 0.4316 - val_acc: 0.8029
Epoch 4/30
5216/5216 [==============================] - 150s 29ms/step - loss: 0.
2263 - acc: 0.9116 - val_loss: 0.8300 - val_acc: 0.6875
Epoch 5/30
5216/5216 [==============================] - 151s 29ms/step - loss: 0.
1905 - acc: 0.9270 - val_loss: 0.6713 - val_acc: 0.7484
Epoch 6/30
5216/5216 [==============================] - 148s 28ms/step - loss: 0.
1617 - acc: 0.9363 - val_loss: 0.6320 - val_acc: 0.7468
Epoch 7/30
5216/5216 [==============================] - 150s 29ms/step - loss: 0.
1485 - acc: 0.9425 - val_loss: 0.8703 - val_acc: 0.7179
Epoch 8/30
5216/5216 [==============================] - 154s 30ms/step - loss: 0.
1330 - acc: 0.9511 - val_loss: 0.6963 - val_acc: 0.7372
Epoch 9/30
5216/5216 [==============================] - 139s 27ms/step - loss: 0.
1253 - acc: 0.9553 - val_loss: 0.7393 - val_acc: 0.7468
Epoch 10/30
5216/5216 [==============================] - 140s 27ms/step - loss: 0.
1172 - acc: 0.9561 - val_loss: 1.0241 - val_acc: 0.7083
```

```
Epoch 11/30
5216/5216 [==============================] - 139s 27ms/step - loss: 0.
1077 - acc: 0.9603 - val_loss: 0.8827 - val_acc: 0.7340
Epoch 12/30
5216/5216 [==============================] - 136s 26ms/step - loss: 0.
0990 - acc: 0.9647 - val_loss: 1.3327 - val_acc: 0.6971
Epoch 13/30
5216/5216 [==============================] - 140s 27ms/step - loss: 0.
0983 - acc: 0.9632 - val_loss: 0.7162 - val_acc: 0.7564
Epoch 14/30
5216/5216 [==============================] - 153s 29ms/step - loss: 0.
0894 - acc: 0.9689 - val_loss: 1.0330 - val_acc: 0.7420
Epoch 15/30
5216/5216 [==============================] - 188s 36ms/step - loss: 0.
0841 - acc: 0.9695 - val_loss: 0.8641 - val_acc: 0.7468
Epoch 16/30
5216/5216 [==============================] - 214s 41ms/step - loss: 0.
0803 - acc: 0.9691 - val_loss: 0.9510 - val_acc: 0.7356
Epoch 17/30
5216/5216 [==============================] - 156s 30ms/step - loss: 0.
0746 - acc: 0.9703 - val_loss: 0.6305 - val_acc: 0.8205
Epoch 18/30
5216/5216 [==============================] - 139s 27ms/step - loss: 0.
0742 - acc: 0.9732 - val_loss: 0.8091 - val_acc: 0.7580
Epoch 19/30
5216/5216 [==============================] - 138s 26ms/step - loss: 0.
0703 - acc: 0.9753 - val_loss: 0.8269 - val_acc: 0.7788
Epoch 20/30
5216/5216 [==============================] - 140s 27ms/step - loss: 0.
0694 - acc: 0.9734 - val_loss: 0.6469 - val_acc: 0.8029
Epoch 21/30
5216/5216 [==============================] - 138s 27ms/step - loss: 0.
0643 - acc: 0.9764 - val_loss: 1.1280 - val_acc: 0.7356
Epoch 22/30
5216/5216 [==============================] - 139s 27ms/step - loss: 0.
0602 - acc: 0.9768 - val_loss: 1.1124 - val_acc: 0.7436
Epoch 23/30
5216/5216 [==============================] - 141s 27ms/step - loss: 0.
0616 - acc: 0.9764 - val_loss: 1.0348 - val_acc: 0.7532
Epoch 24/30
5216/5216 [==============================] - 173s 33ms/step - loss: 0.
0592 - acc: 0.9783 - val_loss: 0.9806 - val_acc: 0.7468
Epoch 25/30
5216/5216 [==============================] - 169s 32ms/step - loss: 0.
0535 - acc: 0.9803 - val_loss: 1.3470 - val_acc: 0.7292
Epoch 26/30
5216/5216 [==============================] - 160s 31ms/step - loss: 0.
0592 - acc: 0.9801 - val_loss: 0.9619 - val_acc: 0.7596
Epoch 27/30
5216/5216 [==============================] - 169s 32ms/step - loss: 0.
0527 - acc: 0.9799 - val_loss: 0.8507 - val_acc: 0.7853
Epoch 28/30
5216/5216 [==============================] - 174s 33ms/step - loss: 0.
0507 - acc: 0.9812 - val_loss: 1.2119 - val_acc: 0.7468
Epoch 29/30
5216/5216 [==============================] - 173s 33ms/step - loss: 0.
0505 - acc: 0.9818 - val_loss: 1.0437 - val_acc: 0.7628
Epoch 30/30
5216/5216 [==============================] - 178s 34ms/step - loss: 0.
0492 - acc: 0.9822 - val_loss: 1.4505 - val_acc: 0.7292
```

In [44]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_6)
```





In [45]:

```
# Evaluate the training results
results_6_train = model_6.evaluate(train_images, train_y)
results_6_train
```

```
5216/5216 [==============================] - 54s 10ms/step
```

Out[45]:

```
[0.05278187918268213, 0.9796779141104295]
```

In [46]:

```
# Evaluate the training results
results_6_test = model_6.evaluate(test_images, test_y)
results_6_test
```

```
624/624 [==============================] - 6s 10ms/step
```

Out[46]:

```
[1.4505285299741304, 0.7291666666666666]
```

3G. Model 7: CNN Model with Data Augmentation

In [47]:

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   zoom_range=0.3,
                                   vertical_flip=True)
```

In [48]:

```
# get all the data in the directory split/train (5216 images), and reshape them
train_generator = train_datagen.flow_from_directory(
        train_folder,
        target_size=(128, 128),
        batch_size = 32,
        class_mode='binary')

# get all the data in the directory split/test (624 images), and reshape them
test_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
        test_folder,
        target_size=(128, 128),
        batch_size = 180,
        class_mode='binary')

# get all the data in the directory split/validation (16 images), and reshape them
val_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
        val_folder,
        target_size=(128, 128),
        batch_size = 32,
        class_mode='binary')
```

```
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

In [49]:

```
model_7 = models.Sequential()
model_7.add(layers.Conv2D(32, (3, 3),
                          activation='relu',
                          input_shape=(128, 128,  3)))
model_7.add(layers.MaxPooling2D((2, 2)))

model_7.add(layers.Conv2D(32, (4, 4), activation='relu'))
model_7.add(layers.MaxPooling2D((2, 2)))

model_7.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_7.add(layers.MaxPooling2D((2, 2)))

model_7.add(layers.Flatten())
model_7.add(layers.Dense(64, activation='relu'))
model_7.add(layers.Dense(1, activation='sigmoid'))

model_7.compile(loss='binary_crossentropy',
                optimizer= 'sgd',
              metrics=['acc'])
```

In [50]:

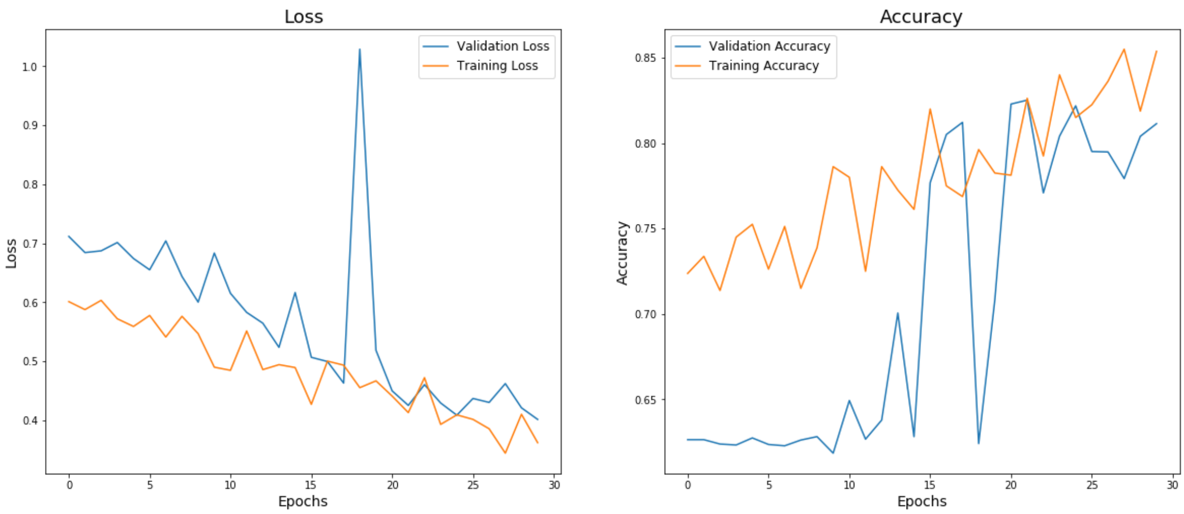
```
results_7 = model_7.fit_generator(train_generator,
                                  steps_per_epoch=25,
                                  epochs=30,
                                  validation_data=test_generator,
                                  validation_steps=25)
```

```
Epoch 1/30
25/25 [==============================] - 140s 6s/step - loss: 0.6011 -
acc: 0.7238 - val_loss: 0.7116 - val_acc: 0.6264
Epoch 2/30
25/25 [==============================] - 137s 5s/step - loss: 0.5877 -
acc: 0.7338 - val_loss: 0.6842 - val_acc: 0.6264
Epoch 3/30
25/25 [==============================] - 155s 6s/step - loss: 0.6033 -
acc: 0.7137 - val_loss: 0.6872 - val_acc: 0.6239
Epoch 4/30
25/25 [==============================] - 148s 6s/step - loss: 0.5721 -
acc: 0.7450 - val_loss: 0.7013 - val_acc: 0.6233
Epoch 5/30
25/25 [==============================] - 112s 4s/step - loss: 0.5591 -
acc: 0.7525 - val_loss: 0.6742 - val_acc: 0.6274
Epoch 6/30
25/25 [==============================] - 120s 5s/step - loss: 0.5777 -
acc: 0.7263 - val_loss: 0.6551 - val_acc: 0.6236
Epoch 7/30
25/25 [==============================] - 97s 4s/step - loss: 0.5412 -
acc: 0.7512 - val_loss: 0.7040 - val_acc: 0.6228
Epoch 8/30
25/25 [==============================] - 106s 4s/step - loss: 0.5763 -
acc: 0.7150 - val_loss: 0.6438 - val_acc: 0.6262
Epoch 9/30
25/25 [==============================] - 119s 5s/step - loss: 0.5469 -
acc: 0.7388 - val_loss: 0.6003 - val_acc: 0.6282
Epoch 10/30
25/25 [==============================] - 88s 4s/step - loss: 0.4902 -
acc: 0.7863 - val_loss: 0.6835 - val_acc: 0.6185
Epoch 11/30
25/25 [==============================] - 124s 5s/step - loss: 0.4849 -
acc: 0.7800 - val_loss: 0.6152 - val_acc: 0.6493
Epoch 12/30
25/25 [==============================] - 130s 5s/step - loss: 0.5515 -
acc: 0.7250 - val_loss: 0.5830 - val_acc: 0.6267
Epoch 13/30
25/25 [==============================] - 117s 5s/step - loss: 0.4861 -
acc: 0.7863 - val_loss: 0.5647 - val_acc: 0.6379
Epoch 14/30
25/25 [==============================] - 116s 5s/step - loss: 0.4944 -
acc: 0.7725 - val_loss: 0.5238 - val_acc: 0.7006
Epoch 15/30
25/25 [==============================] - 111s 4s/step - loss: 0.4896 -
acc: 0.7612 - val_loss: 0.6166 - val_acc: 0.6282
Epoch 16/30
25/25 [==============================] - 106s 4s/step - loss: 0.4272 -
acc: 0.8200 - val_loss: 0.5069 - val_acc: 0.7769
Epoch 17/30
25/25 [==============================] - 113s 5s/step - loss: 0.5004 -
acc: 0.7750 - val_loss: 0.4997 - val_acc: 0.8050
Epoch 18/30
25/25 [==============================] - 101s 4s/step - loss: 0.4936 -
```

```
acc: 0.7688 - val_loss: 0.4632 - val_acc: 0.8122
Epoch 19/30
25/25 [==============================] - 97s 4s/step - loss: 0.4555 -
acc: 0.7963 - val_loss: 1.0287 - val_acc: 0.6241
Epoch 20/30
25/25 [==============================] - 135s 5s/step - loss: 0.4670 -
acc: 0.7825 - val_loss: 0.5187 - val_acc: 0.7085
Epoch 21/30
25/25 [==============================] - 110s 4s/step - loss: 0.4412 -
acc: 0.7812 - val_loss: 0.4500 - val_acc: 0.8229
Epoch 22/30
25/25 [==============================] - 109s 4s/step - loss: 0.4133 -
acc: 0.8263 - val_loss: 0.4254 - val_acc: 0.8252
Epoch 23/30
25/25 [==============================] - 100s 4s/step - loss: 0.4724 -
acc: 0.7925 - val_loss: 0.4606 - val_acc: 0.7709
Epoch 24/30
25/25 [==============================] - 99s 4s/step - loss: 0.3933 -
acc: 0.8400 - val_loss: 0.4295 - val_acc: 0.8041
Epoch 25/30
25/25 [==============================] - 109s 4s/step - loss: 0.4094 -
acc: 0.8150 - val_loss: 0.4088 - val_acc: 0.8219
Epoch 26/30
25/25 [==============================] - 112s 4s/step - loss: 0.4018 -
acc: 0.8225 - val_loss: 0.4372 - val_acc: 0.7951
Epoch 27/30
25/25 [==============================] - 121s 5s/step - loss: 0.3858 -
acc: 0.8362 - val_loss: 0.4305 - val_acc: 0.7949
Epoch 28/30
25/25 [==============================] - 113s 5s/step - loss: 0.3445 -
acc: 0.8550 - val_loss: 0.4624 - val_acc: 0.7793
Epoch 29/30
25/25 [==============================] - 87s 3s/step - loss: 0.4104 -
acc: 0.8188 - val_loss: 0.4213 - val_acc: 0.8040
Epoch 30/30
25/25 [==============================] - 129s 5s/step - loss: 0.3623 -
acc: 0.8538 - val_loss: 0.4017 - val_acc: 0.8114
```

In [51]:

```
# Visualise the loss and accuracy of the training and validation sets across epochs
visualize_results(results_7)
```



In [52]:

```
train_x, train_y = next(train_generator)
```

In [53]:

```
# Evaluate the training results
results_7_train = model_7.evaluate(train_x, train_y)
results_7_train
```

```
32/32 [==============================] - 1s 19ms/step
```

Out[53]:

```
[0.1908160299062729, 0.9375]
```

In [54]:

```
test_x, test_y = next(test_generator)
```

In [55]:

```
# Evaluate the training results
results_7_test = model_7.evaluate(test_x, test_y)
results_7_test
```

```
180/180 [==============================] - 2s 12ms/step
```

Out[55]:

```
[0.3883267707294888, 0.8111111137602064]
```

4. Final Model Performance Evaluation

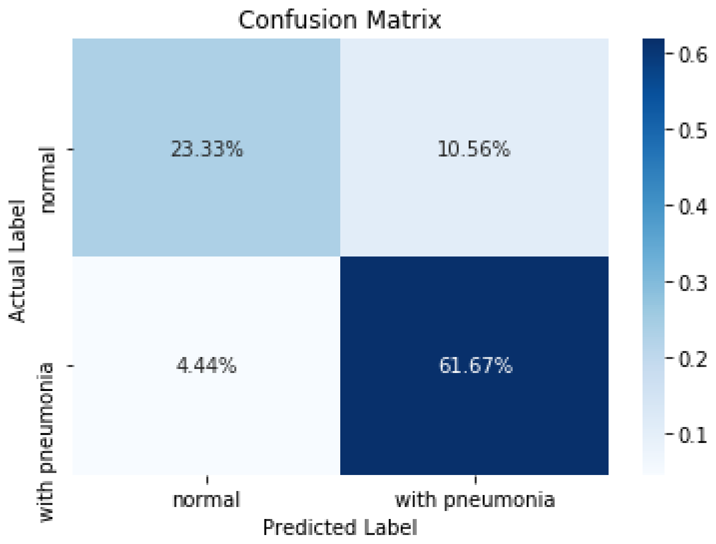
In [70]:

```
# Import necessary libraries for performance evaluation.
from sklearn.metrics import accuracy_score, confusion_matrix

# Create predictions
preds = model_7.predict(test_x)

# Calculate accuracy and confusion matrix
acc = accuracy_score(test_y, np.round(preds))*100
cm = confusion_matrix(test_y, np.round(preds))
tn, fp, fn, tp = cm.ravel()

print('CONFUSION MATRIX ------------------')
print(cm)

print('\nTEST METRICS ----------------------')
precision = tp/(tp+fp)*100
recall = tp/(tp+fn)*100
print('Accuracy: {}%'.format(acc))
print('Precision: {}%'.format(precision))
print('Recall: {}%'.format(recall))
print('F1-score: {}'.format(2*precision*recall/(precision+recall)))

print('\nTRAIN METRIC ----------------------')
print('Train acc: {}'.format(np.round((results_7.history['acc'][-1])*100, 2)))
```

```
CONFUSION MATRIX ------------------
[[ 42  19]
 [  8 111]]

TEST METRICS ----------------------
Accuracy: 85.0%
Precision: 85.38461538461539%
Recall: 93.27731092436974%
F1-score: 89.1566265060241

TRAIN METRIC ----------------------
Train acc: 85.38
```

In [82]:

```
# Import Seaborn and make the confusion matrix more visually presentable
import seaborn as sns

ax = plt.subplot()
sns.heatmap(cm/np.sum(cm), annot=True, ax=ax, fmt='.2%', cmap='Blues')

ax.set_title('Confusion Matrix')
ax.set_xlabel("Predicted Label")
ax.set_ylabel("Actual Label")
ax.xaxis.set_ticklabels(['normal', 'with pneumonia'])
ax.yaxis.set_ticklabels(['normal', 'with pneumonia'])
plt.show();
```



Conclusion

Initial models did not generalise well and tended to overfit the training data. Subsequent changes to parameters had little or no effect with significant fluctuations in accuracy against validation data observed.

Augmentation of the training data by flipping the x-ray images vertically struck a better balance between training and validation accuracy.

Whilst it does not have a high accuracy score, this final model using a Convolutional Neural Network and data augmentation produced the best results classifying 85% of the “unseen” chest x-ray images, but there is a significant resource overhead when using, taking over an hour to train the model.

Further manipulation of the architecture of Convolutional Neural Network could also potentially improve accuracy.