NTHU CS4602 HW3

COVID19 30-day Mortality Prediction from CXR

KUO, KUAN-TING

Contents

Abstract

This is the third assignment for CS4602.
By using the chest X-ray of 1393 patients and their medical records, the mortality of these patients after 30 days are predicted.

Dataset Information

Training Data

- Number of patients: 1393 in total
    - 1229 in class 0 and 164 in class 1
- Data per patient:
    - A 320x320 chest X-ray image
    - Medical records with 47 attributes stored in a CSV

Testing Data

- 457 patients with X-ray images and medical records

Model

The entire model pipeline contains two steps.
First, the image files are trained with DenseNet 201, which is provided in the Keras API (https://keras.io/api/applications/), in a transfer learning approach. The predictions are then stored in a CSV file - 107062274.csv.
Then, the CSV file with the medical records and the image prediction CSV file are joined together to a CSV file.
Finally, an SVM model, which is provided from sklearn, would be trained based on the above-mentioned CSV file, and the new prediction result would be store to a new CSV file - Bonus_107062274.csv.

Data Preprocess

Different methods are used to do the preprocessing for the two data types.

The medical record CSV file

- Missing information: Filled the most frequent values for the missing categorical features and median for the missing numerical features.

- One hot transformation: Transfer text-like attributes that are not trainable into one hot columns.

  ```
  data_dum = pd.get_dummies(df, prefix=['s', 'd'], columns=['sex',
  'ed_diagnosis'])
  ```

Image files

- Preprocess for model: Different models needed different preprocess input command for loading the input images. In this study, DenseNet 201 is chosen.

  ```
  x = keras.applications.densenet.preprocess_input(inputs)
  ```

- Data augmetation: Use (-0.01, 0.01) x 360 degree to randomly rotate data for augmentation.

  ```
  x = keras.layers.experimental.preprocessing.RandomRotation((-0.01,
  0.01))(x)
  ```

Selected Model Design

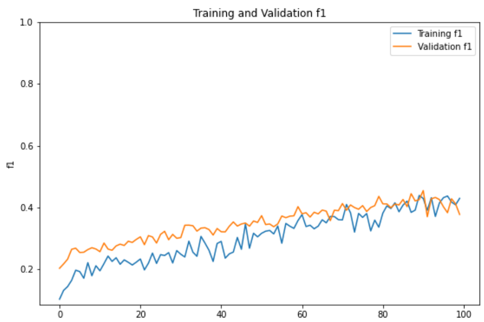
The final results are predicted based on the following model design.

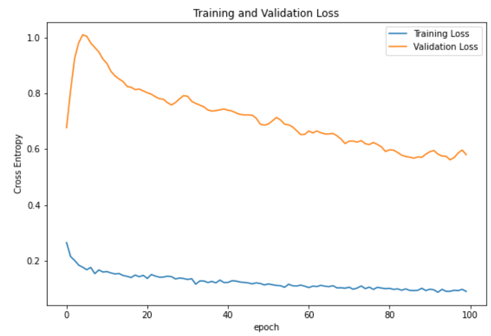
107062274_HW3_Model

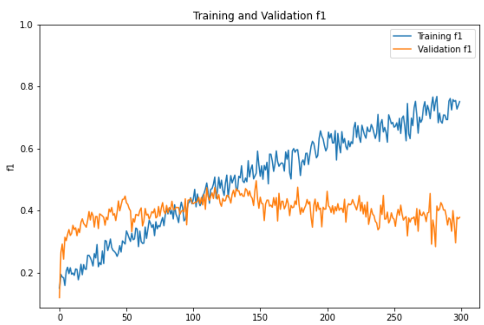
- Input: 320x320 chest X-ray images
- Output: Mortality prediction with [0, 1] values
- Method: Transfer learning
  - Base model: DenseNet 201 with the last 20 layers set as trainable.
  - Additional layers: 3 Dense layers with activation='relu' followed by different drop out values

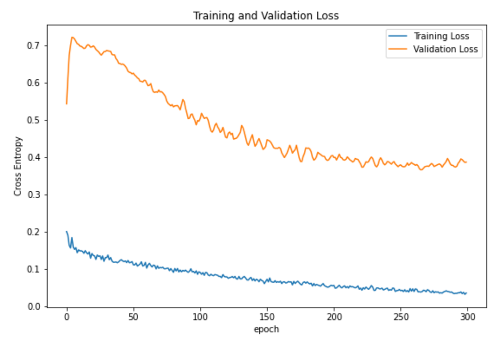
```
inputs = keras.Input(shape=IMG_SHAPE)

x = keras.applications.densenet.preprocess_input(inputs)

x = keras.layers.experimental.preprocessing.RandomRotation((-0.01,
0.01))(x)

x = base_model(x, training=False)

x = keras.layers.Flatten()(x)

x = keras.layers.Dense(512, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(256, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(64, activation='relu')(x)

x = keras.layers.Dropout(0.4)(x)

outputs = keras.layers.Dense(1, activation='sigmoid')(x)

model = keras.Model(inputs, outputs)
```

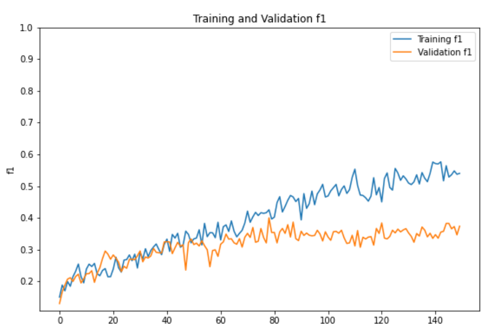
- Below are the training and validation performances using different training and validation dataset combinations.

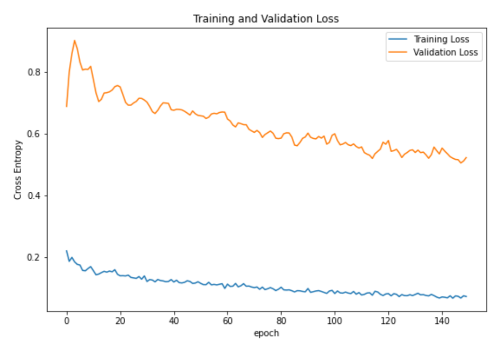












Bonus_107062274_HW3_Model

- Classification model: SVM was chosen after comparing the performances of other classification models, including RandomForest, Naive Bayes, Decision Tree, and AdaBoost.
- StandardScaler: In the model pipeline, StandardScaler is applied to normalized the input to range in [-1, 1].
- SVM kernel: After trying linear, poly, sigmoid, and RBF, sigmoid kernels are used because it outputs the most stable f1 scores.
- C and gamma tunning: Grid search algorithm is used here as a kind of greedy search to optimize the value of C and gamma.
-

Approach Comparison

Dimension Reduction

I tried using different dimension reduction methods to reduce the dimension of the image data and train them on an SVM model.
The f1 scores are below 0.25 on average.
Therefore, I decided to turn to another method - transfer learning.

Transfer Learning

There are many base models selectable in Keras Applications (as shown as below).
I tried various combination. The Densenet models turned out to have the most stable and well-performed results. Therefore, Densenet201 is used in this study.

Experiment and Results

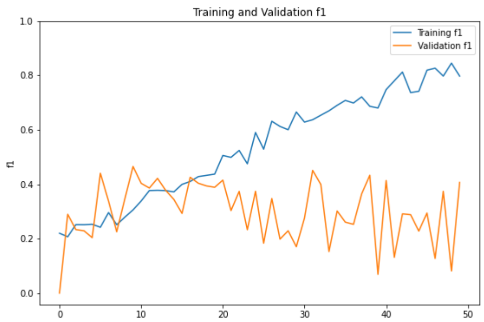
I tried tuning the parameters in this model in various ways. Below are some experiments that are done along with the results.

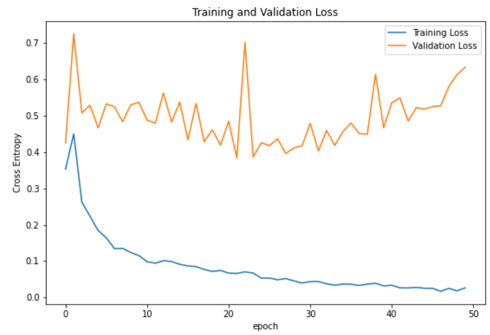
Three additional Dense layers (activation = 'relu')

```
x = keras.layers.Dense(512, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(256, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(64, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)
```

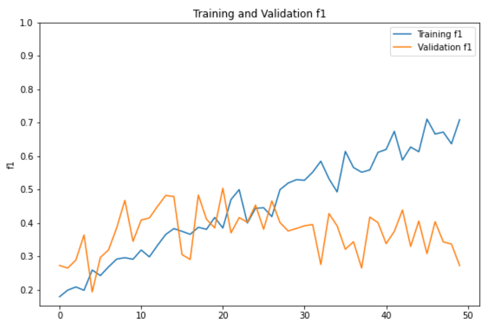
Different Learning Rate (last 15 DenseNet201 layers trainable)

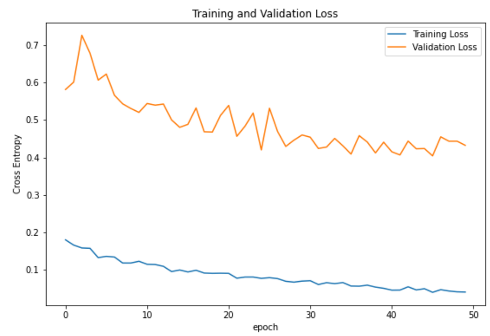
- Basic setting: class weighted 1:9 and data augmentation with RandomRotation((-0.01, 0.01))
- Learning Rate: 0.0001



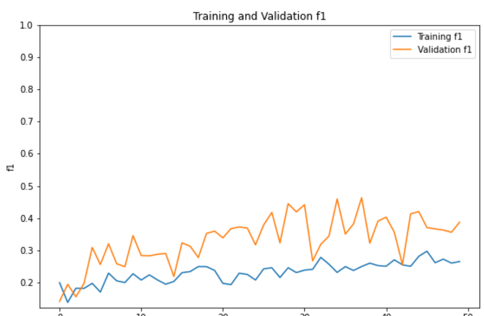


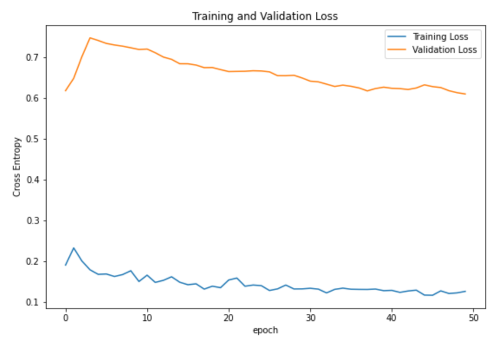
- Learning Rate: 0.00001





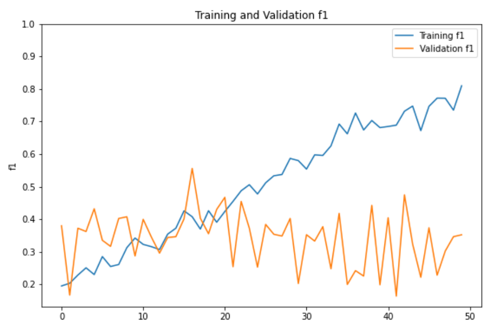
- Learning Rate: 0.000001

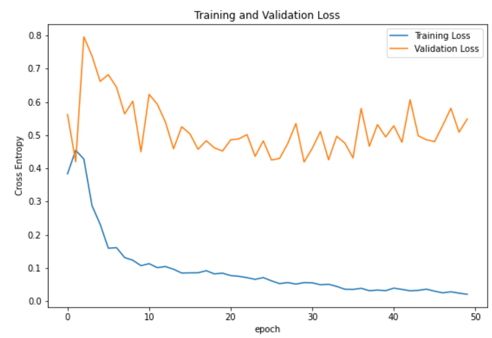




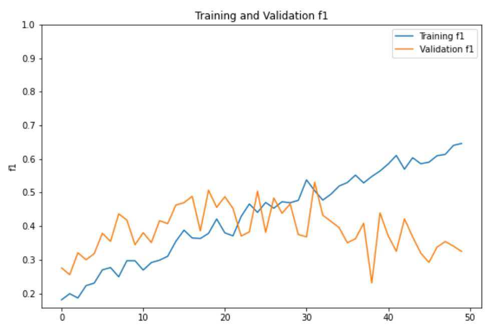
Different Learning Rate Comparison (last 30 DenseNet201 layers trainable)

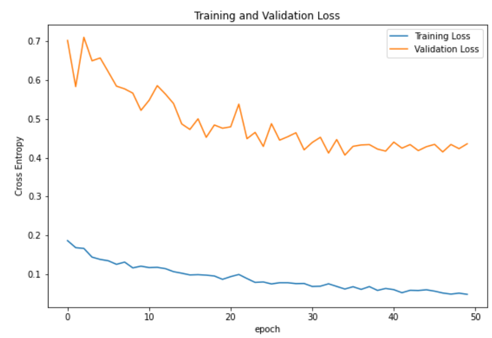
- Basic setting: class weighted 1:9 and data augmentation with RandomRotation((-0.01, 0.01))
- Learning Rate: 0.0001



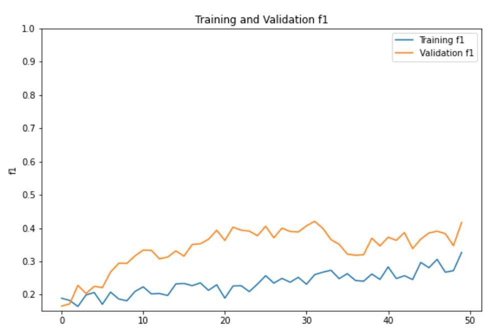


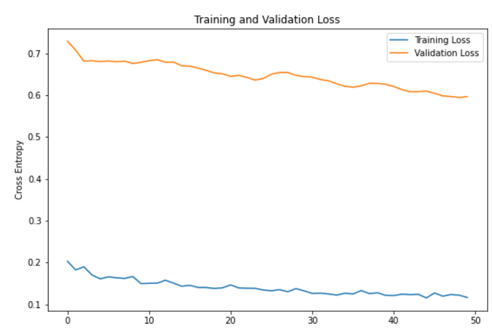
- Learning Rate: 0.00001





- Learning Rate: 0.000001



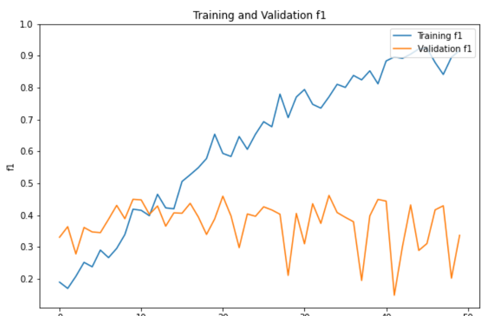


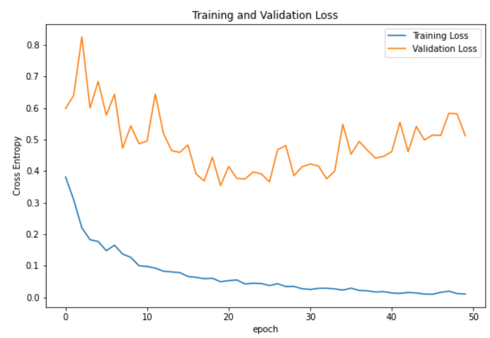
Two additional Dense layers (activation = 'relu')

```
x = keras.layers.Dense(512, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)

x = keras.layers.Dense(64, activation='relu')(x)

x = keras.layers.Dropout(0.5)(x)
```

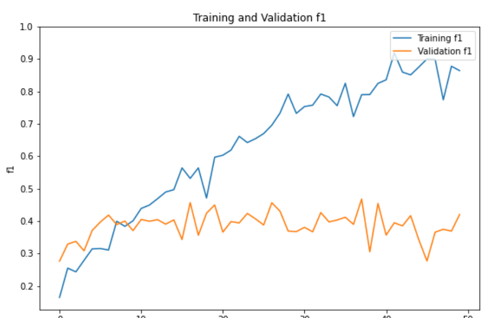
Different Learning Rate Comparison (last 30 DenseNet201 layers trainable)

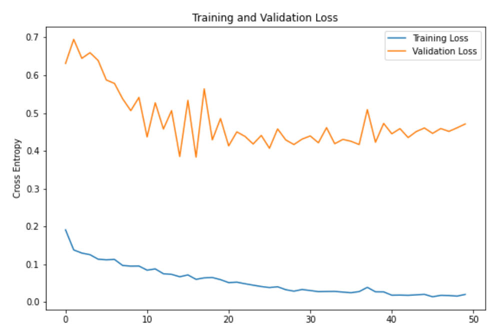
- Basic setting: class weighted 1:9 and data augmentation with RandomRotation((-0.01, 0.01))
- Learning Rate: 0.0001



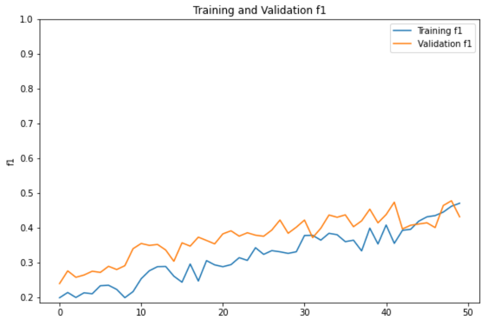


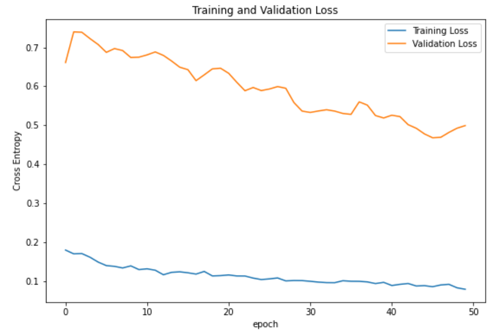
- Learning Rate: 0.00001

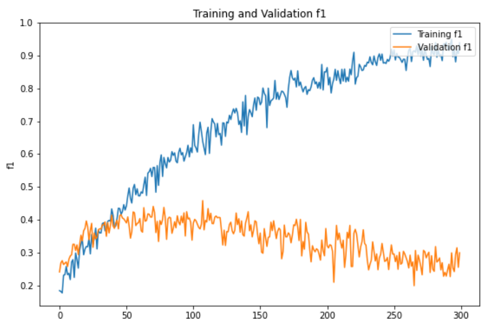


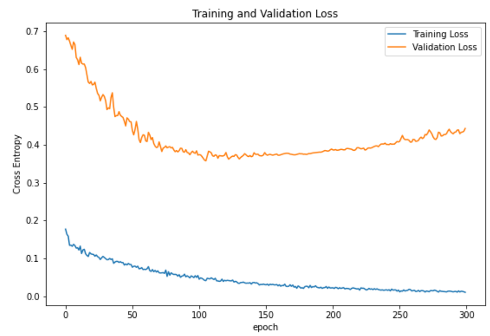


- Learning Rate: 0.000001



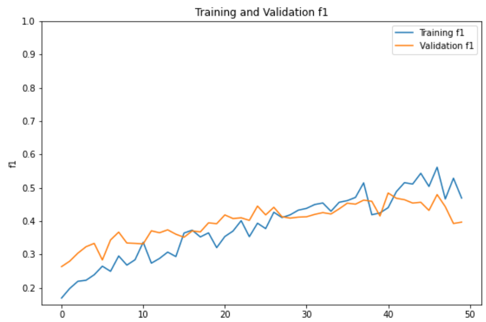


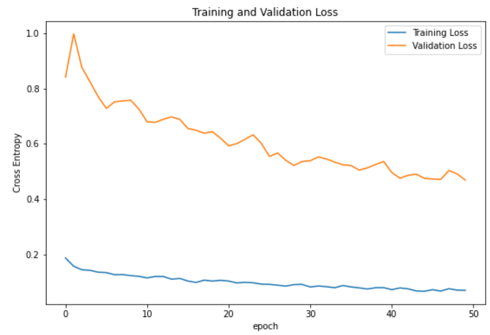


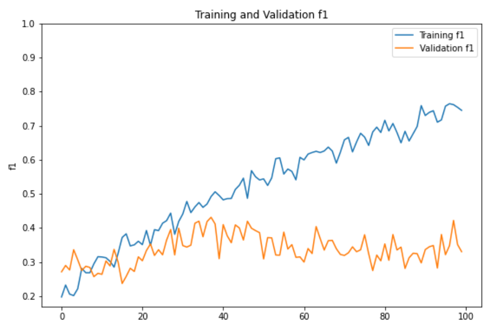


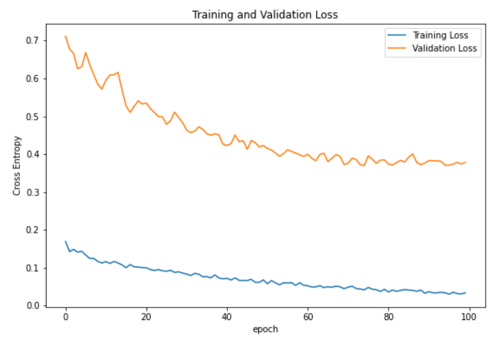
Different Dropout Comparison (last 30 DenseNet201 layers trainable)

- Basic setting: Learning Rate: 0.000001/ Class weighted 1:9/ Data augmentation with RandomRotation((-0.01, 0.01))/ Two dense layers
- Dropout:0.5/ 0.3

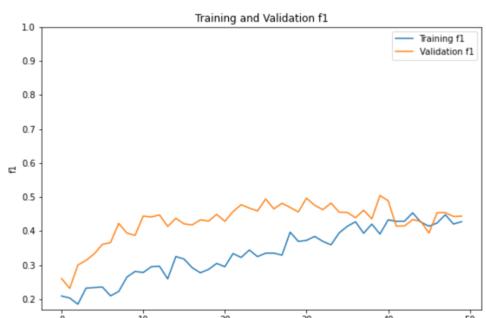


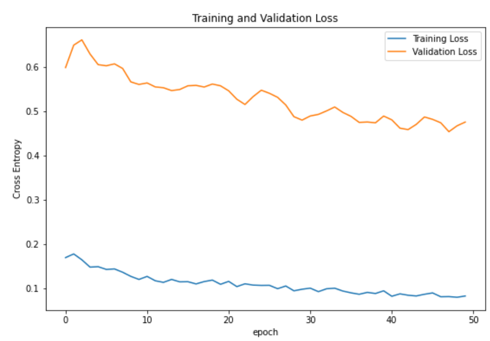


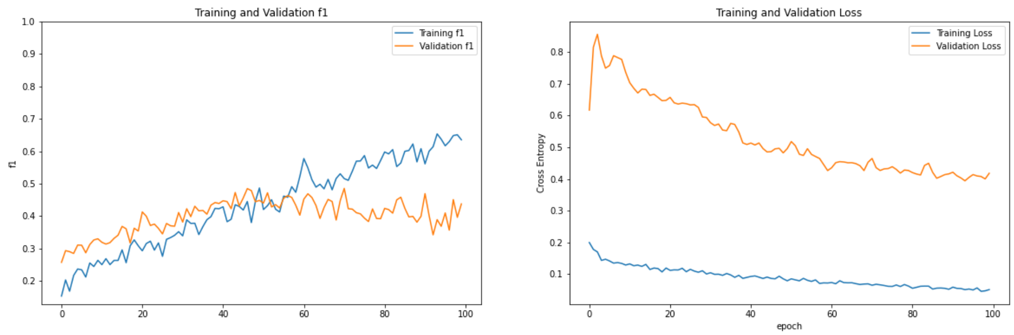




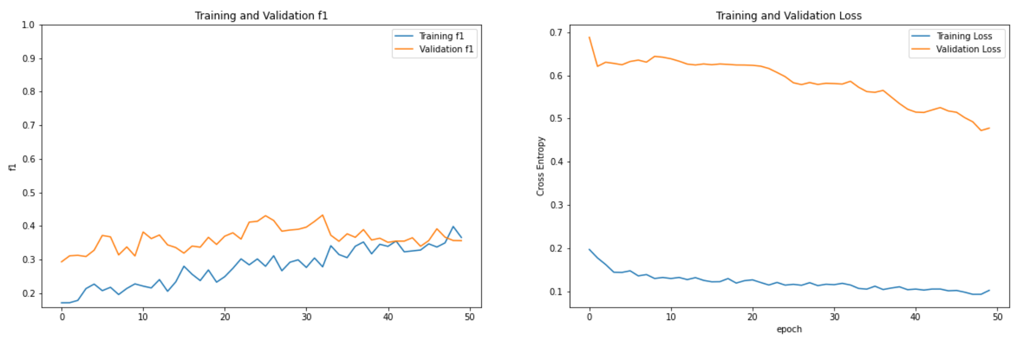
- Dropout:0.5/ 0.5



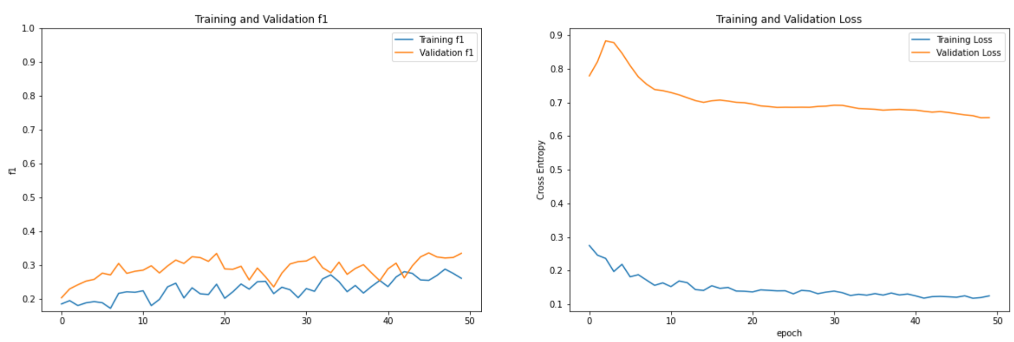




- Dropout:0.5/ 0.6



- Dropout:0.5/ 0.8



Reference Link

https://www.tensorflow.org/tutorials/images/transfer_learning