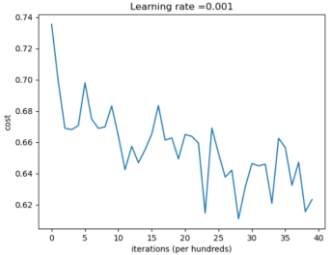
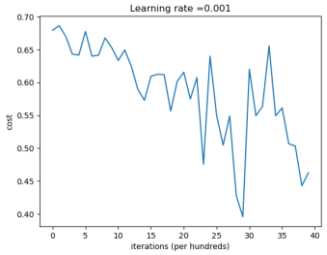
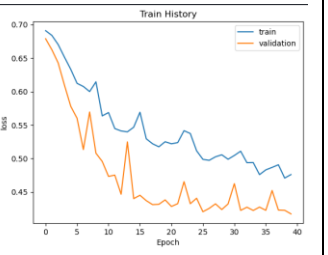


Training size = 480, validation size = 120

	Linear model	CNN model	TensorFlow CNN model
epochs	40	40	40
training time	0.5s	49m 1.5s	4.8s
Train accuracy	0.76	0.75	0.80
valid accuracy	0.8	0.84	0.84
num of parameters	65665	39473	39473
training loss curve			

雖然 linear 的 train accuracy 比 CNN model 高，但是他的 loss 比較高，而 tensorflow 的表現最好，loss 比較不會上上下下，並且 training time 比 CNN model 快很多。

Advanced part:

```
1 model = models.Sequential()
2 # model.add(layers.experimental.preprocessing.RandomFlip("horizontal"))
3
4 model.add(layers.Conv2D(filters=16,
5                         kernel_size=(3, 3),
6                         strides=(2, 2),
7                         activation='relu'))
8
9 model.add(layers.MaxPool2D(pool_size=(2, 2), strides=1))
10
11 model.add(layers.Conv2D(filters=16,
12                        kernel_size=(3, 3),
13                        strides=(2, 2),
14                        activation='relu'))
15
16 model.add(layers.Dropout(0.25))
17
18 model.add(layers.Flatten())
19
20 model.add(layers.Dense(units=64, activation='relu'))
21 model.add(layers.Dense(units=1, activation='sigmoid'))
22
23 model.compile(optimizer=tf.keras.optimizers.Adam(),
24              loss=tf.keras.losses.BinaryCrossentropy(),
25              metrics=['accuracy'])
```

我先使用了 Conv2D、MaxPool2D、Conv2D 後，多加了一層的 Dropout 來避免 overfitting，後面再進行 Flatten 和 Dense layer，原本還有使用 data augmentation，但是發現表現沒有差很多，所以最後就移掉了。

Loss function 因為是 class 數是 2 所以一樣使用 BinaryCrossEntropy，Optimizer 選擇 Adam，因為它可以比較好的調整 learning rate。

```
34 train_history = model.fit(X_train, y_train,
35                          # validation_data=(X_val, y_val),
36                          batch_size=64,
37                          epochs=100,
38                          verbose=1
39                          )
```

Batch size 為 40，epochs 做 100 次