AI 無線通訊系統實驗

Fianl Project

賴昱凱 111511141

一、資料前處理

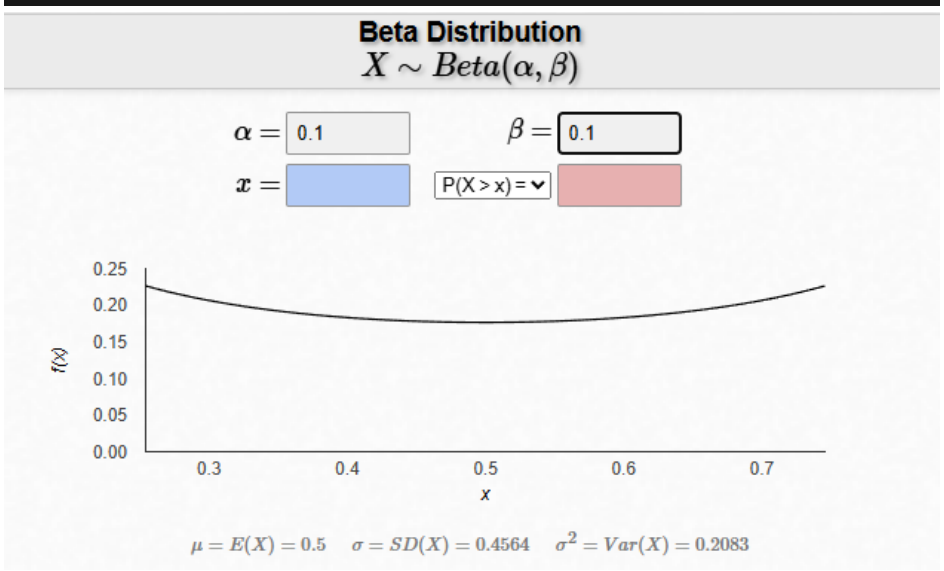
1. 選哪根天線

```
rx_size = 8;
tx_size = 8;
data_tx_rx = data_reshape(:,9:16,5:12);
```

2. Data augmentation

隨機取兩筆資料做線性合併(input, output 都是，可以模擬出更多種反射錐擺放位置)，讓模型更容易真正學會資料的意義，可以大幅增加模型的泛化能力。然而 alpha = 1 時理論上資料會是最豐富的，因此泛化能力應該增加最多，但實際訓練後發現，雖然確實 training acc 與 validation acc 的差距有明顯的減小，但反而因為太多在 output 在 0.5 附近的答案，讓模型無法那麼精確的剛好有兩個 1 其他都是 0，因此 training acc 本身就無法太高。因此在多次實驗後，使用 alpha = 0.1 做訓練是最佳的選擇。

```python
def mixup_tf(x, y, alpha=0.1):
    lam = tfp.distributions.Beta(alpha, alpha).sample()
    index = tf.random.shuffle(tf.range(tf.shape(x)[0]))
    x_mix = lam * x + (1 - lam) * tf.gather(x, index)
    y_mix = lam * y + (1 - lam) * tf.gather(y, index)
    return x_mix, y_mix
```



二、CNN 架構

```
model = keras.Sequential([
    keras.layers.Conv2D(128, (3, 3), input_shape=(train_input.shape[1], train_input.shape[2], 2)),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Conv2D(128, (3, 3), padding='same'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),

    keras.layers.Conv2D(256, (3, 3), padding='same'),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Conv2D(256, (3, 3), padding='same'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),

    keras.layers.Conv2D(512, (3, 3), padding='same'),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Conv2D(512, (3, 3), padding='same'),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),

    keras.layers.Conv2D(1024, (3, 3), padding='same'),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Conv2D(1024, (3, 3), padding='same'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
```

```
    keras.layers.GlobalAveragePooling2D(),

    keras.layers.Dense(512),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Dropout(config['dropout_rate']),
    keras.layers.Dense(1024),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Dropout(config['dropout_rate']),
    keras.layers.Dense(128),
    keras.layers.BatchNormalization(),
    keras.layers.LeakyReLU(alpha=0.3),
    keras.layers.Dropout(config['dropout_rate']),
    keras.layers.Dense(train_output.shape[1]),
    keras.layers.Activation('sigmoid'),
])
```

三、Loss function

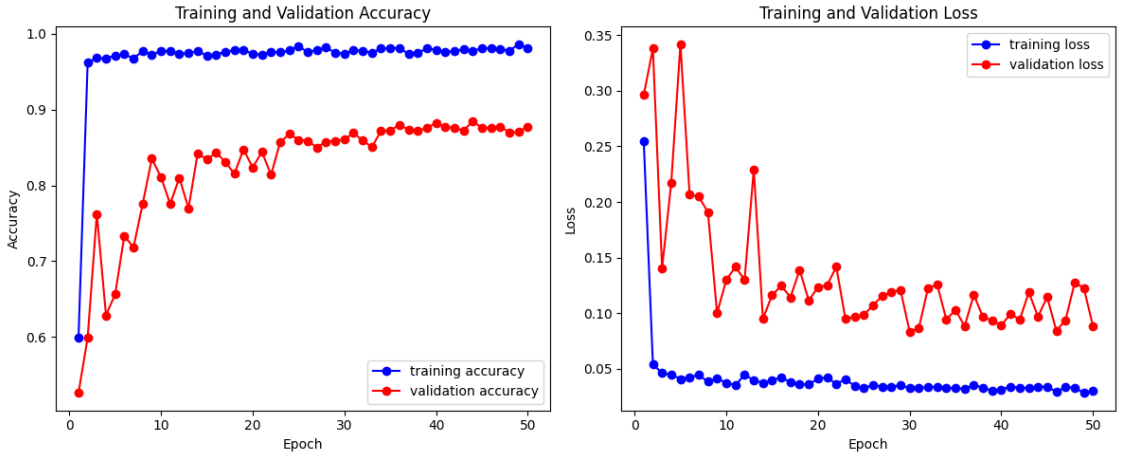
loss = tf.keras.losses.BinaryCrossentropy(from_logits=False),

四、正確率的計算

```python
def exact_match_accuracy(y_true, y_pred):
    y_pred_binary = tf.cast(y_pred > 0.5, tf.float32)
    y_true_binary = tf.cast(y_true > 0.5, tf.float32)

    # 比對是否逐 row 全部正確
    match = tf.reduce_all(tf.equal(y_true_binary, y_pred_binary), axis=1)

    # 計算準確率
    accuracy = tf.reduce_mean(tf.cast(match, tf.float32))

    return accuracy
```

五、最終結果

我是把 test data 放在 validation data 方便觀察訓練過程，但沒有針對 validation 做任何訓練上的改變或是對訓練過程有所影響(包括 early stop、根據 validation loss 的 lr 改變等都沒有)，所以不會有作弊的問題。




```
model.evaluate(test_dataset, verbose=1)
```

```
112/112 [==============================] - 3s 28ms/step - loss: 0.0882 - exact_match_accuracy: 0.8777
[0.08815385401248932, 0.8777204155921936]
```

Accuracy: 87.78%