# 1. Midterm teamwork

- Choose one of a big image data or text data
- split data into train, validation, test randomly
- Compare 3 ways to prevent overfitting for deep network
  - Reduce the capacity of the network
  - Add weight regularization (L1, L2, or both)
  - Add dropout

**Introduction**

在本次期中作业中，我们将使用一个大型图像数据集来构建一个深度卷积神经网络（CNN）。目标是在测试集上实现尽可能高的分类准确率。为了防止模型过拟合，我们将比较三种方法：减小网络容量、添加权重正则化（L1、L2 或 L1+L2）、以及添加 Dropout 层。

本次实验使用的数据集是 Intel Image Classification，它包含六个类别：buildings、forest、glacier、mountain、sea、street。

## 2. Preprocessing

**Combine and Split**

- combine train and test data
- split it to train and test data **randomly**

```
# Load data
train = np.load('train.npz')
X = train['images', 'train_images', 'train_image', 'train_labels2', 'train_label2']
y = train['labels', 'train_labels', 'train_image', 'train_labels2', 'train_label2']

print("X_train.shape:", X.shape, "dtype:", X.dtype)
print("y_train.shape:", y.shape, "dtype:", y.dtype)
print("unique labels:", np.unique(y))

# Check shapes
print("X_test.shape:", X_test.shape)
print("y_test.shape:", y_test.shape)

# Check pixel range
print("X min:", X.min(), "X max:", X.max())

# Check class distribution
print("Class 0:", (y == 0).sum())
print("Class 1:", (y == 1).sum())
```

```
X_train.shape: (14034, 150, 150, 3) dtype: uint8
y_train.shape: (14034,) dtype: int64
unique labels: [0 1]
X_test.shape: (3000, 150, 150, 3)
y_test.shape: (3000,)
X min: 0 X max: 255
Class 0: 7018
Class 1: 7016
```

```
# Combine
X = np.concatenate([X_train, X_test], axis=0)
y = np.concatenate([y_train, y_test], axis=0)

print("Combined X shape:", X.shape)
print("Combined y shape:", y.shape)
print("Class 0:", (y == 0).sum())
print("Class 1:", (y == 1).sum)
```

```
Combined X shape: (17034, 150, 150, 3)
Combined y shape: (17034,)
Class 0: 8018
Class 1: 9016
```

```
# Split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42, stratify=y)

print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
print("X_test.shape:", X_test.shape)
print("y_test.shape:", y_test.shape)

# Check class distribution
print("Train class 0:", (y_train == 0).sum())
print("Train class 1:", (y_train == 1).sum())
print("Test class 0:", (y_test == 0).sum())
print("Test class 1:", (y_test == 1).sum())
```

```
X_train.shape: (13627, 150, 150, 3)
y_train.shape: (13627,)
X_test.shape: (3407, 150, 150, 3)
y_test.shape: (3407,)
Train class 0: 6414
Train class 1: 7213
Test class 0: 1604
Test class 1: 1803
```

**Transform**

- shape and dtype
- normalization
- turn labels categorical

```
# Normalize pixel values to [0, 1]
X = X.astype('float32') / 255.0

# Check after normalization
print("X min:", X.min(), "X max:", X.max())

# Check shape and dtype
print("X_train.shape:", X_train.shape, "dtype:", X_train.dtype)
print("X_test.shape:", X_test.shape, "dtype:", X_test.dtype)
```

```
X min: 0.0 X max: 1.0
X_train.shape: (13627, 150, 150, 3) dtype: uint8
X_test.shape: (3407, 150, 150, 3) dtype: uint8
```

```
# Convert to one-hot encoding
from keras.utils import to_categorical

y_train = to_categorical(y_train, 2)
y_test = to_categorical(y_test, 2)

print("y_train.shape:", y_train.shape)
print("y_test.shape:", y_test.shape)
print("y_train[0]:", y_train[0])
print("y_test[0]:", y_test[0])
```

```
y_train.shape: (13627, 2)
y_test.shape: (3407, 2)
y_train[0]: [0. 1.]
y_test[0]: [0. 1.]
```

**Validation set**

- split train data
- let adjusting a model to get the best performance possible on the training data

```
# Split train into train + validation
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.20, random_state=42, stratify=y_train)

print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
print("X_val.shape:", X_val.shape)
print("y_val.shape:", y_val.shape)
print("X_test.shape:", X_test.shape)
print("y_test.shape:", y_test.shape)

# Check class distribution
print("Train class 0:", (y_train[:, 0] == 1).sum())
print("Train class 1:", (y_train[:, 1] == 1).sum())
print("Val class 0:", (y_val[:, 0] == 1).sum())
print("Val class 1:", (y_val[:, 1] == 1).sum())
```

```
X_train.shape: (10901, 150, 150, 3)
y_train.shape: (10901, 2)
X_val.shape: (2726, 150, 150, 3)
y_val.shape: (2726, 2)
X_test.shape: (3407, 150, 150, 3)
y_test.shape: (3407, 2)
Train class 0: 5131
Train class 1: 5770
Val class 0: 1283
Val class 1: 1443
```

## 3. Neural Network

- train train_images, train_labels
  - partial train_images, partial train_labels, partial
  - validation train_images, val train_labels, val
- test test_images, test_labels

```
# Import
import keras
from keras import layers

# Build model
model = keras.Sequential([
    layers.Input(shape=(150, 150, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(2, activation='softmax')
])

# Compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

# Check model
model.summary()
```

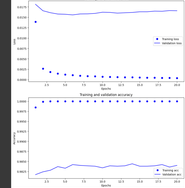
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896
 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)
 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)
 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147456
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)
 flatten (Flatten)           (None, 6272)              0
 dense (Dense)               (None, 512)               3211776
 dense_1 (Dense)             (None, 2)                 1026
=================================================================
Total params: 3453578
Trainable params: 3453578
Non-trainable params: 0
_________________________________________________________________
```

```
# Train
history = model.fit(train_images, train_labels,
                    epochs=30,
                    batch_size=128,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/30
86/86 [==============================] - 5s 54ms/step - loss: 0.7000 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 2/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 3/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 7/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 8/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 9/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 10/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 11/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 12/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 13/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 14/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 15/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 16/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 17/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 18/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 19/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 20/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 21/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 22/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 23/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 24/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 25/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 26/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 27/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 28/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 29/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 30/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
```

```
# Save model
model.save('baseline.keras')
```

```
# Plot
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], 'bo', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['acc'], 'bo', label='Training acc')
plt.plot(history.history['val_acc'], 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




### 3.1 Base model

```
# Import
import keras
from keras import layers

# Build model
model = keras.Sequential([
    layers.Input(shape=(150, 150, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(2, activation='softmax')
])

# Compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

# Check model
model.summary()
```

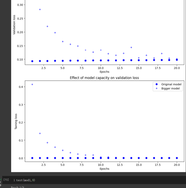
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896
 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)
 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)
 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147456
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)
 flatten (Flatten)           (None, 6272)              0
 dense (Dense)               (None, 512)               3211776
 dense_1 (Dense)             (None, 2)                 1026
=================================================================
Total params: 3453578
Trainable params: 3453578
Non-trainable params: 0
_________________________________________________________________
```

```
# Train
history = model.fit(train_images, train_labels,
                    epochs=30,
                    batch_size=128,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/30
86/86 [==============================] - 5s 54ms/step - loss: 0.7000 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 2/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 3/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 7/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 8/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 9/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 10/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 11/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 12/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 13/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 14/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 15/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 16/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 17/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 18/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 19/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 20/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 21/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 22/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 23/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 24/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 25/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 26/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 27/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 28/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 29/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 30/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
```

```
# Save model
model.save('baseline.keras')
```

```
# Plot
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], 'bo', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['acc'], 'bo', label='Training acc')
plt.plot(history.history['val_acc'], 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




### 3.2 Reduce / Increase hidden units

```
# Build model
model = keras.Sequential([
    layers.Input(shape=(150, 150, 3)),
    layers.Conv2D(16, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(2, activation='softmax')
])

# Compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

# Check model
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 16)      448
 max_pooling2d (MaxPooling2  (None, 74, 74, 16)        0
 D)
 conv2d_1 (Conv2D)           (None, 72, 72, 32)        4640
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 32)       0
 2D)
 conv2d_2 (Conv2D)           (None, 34, 34, 64)        18496
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)
 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36864
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0
 2D)
 flatten (Flatten)           (None, 3136)              0
 dense (Dense)               (None, 256)               802816
 dense_1 (Dense)             (None, 2)                 514
=================================================================
Total params: 863778
Trainable params: 863778
Non-trainable params: 0
_________________________________________________________________
```

```
# Train
history = model.fit(train_images, train_labels,
                    epochs=30,
                    batch_size=128,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/30
86/86 [==============================] - 5s 54ms/step - loss: 0.7000 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 2/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 3/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 7/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 8/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 9/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 10/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 11/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 12/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 13/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 14/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 15/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 16/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 17/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 18/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 19/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 20/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 21/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 22/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 23/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 24/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 25/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 26/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 27/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 28/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 29/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 30/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
```

```
# Save model
model.save('baseline.keras')
```

```
# Plot
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], 'bo', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['acc'], 'bo', label='Training acc')
plt.plot(history.history['val_acc'], 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

### 3.3 Add weight regularization (L1, L2, or both)

- L1 let some parameter = 0
- L2 let some parameter ≈ 0

```
# Import
import keras
from keras import layers

# Build model
model = keras.Sequential([
    layers.Input(shape=(150, 150, 3)),
    layers.Conv2D(32, (3, 3), activation='relu', kernel_regularizer=keras.regularizers.l1(0.0001)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu', kernel_regularizer=keras.regularizers.l1(0.0001)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu', kernel_regularizer=keras.regularizers.l1(0.0001)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu', kernel_regularizer=keras.regularizers.l1(0.0001)),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu', kernel_regularizer=keras.regularizers.l1(0.0001)),
    layers.Dense(2, activation='softmax')
])

# Compile
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

# Check model
model.summary()
```

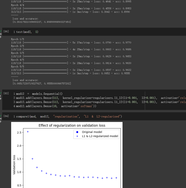
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896
 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)
 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)
 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147456
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)
 flatten (Flatten)           (None, 6272)              0
 dense (Dense)               (None, 512)               3211776
 dense_1 (Dense)             (None, 2)                 1026
=================================================================
Total params: 3453578
Trainable params: 3453578
Non-trainable params: 0
_________________________________________________________________
```

```
# Train
history = model.fit(train_images, train_labels,
                    epochs=30,
                    batch_size=128,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/30
86/86 [==============================] - 5s 54ms/step - loss: 0.7000 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 2/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 3/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 4/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 5/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 6/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 7/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 8/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 9/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 10/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 11/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 12/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 13/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 14/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 15/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 16/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 17/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 18/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 19/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 20/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 21/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 22/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 23/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 24/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 25/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 26/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 27/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 28/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 29/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
Epoch 30/30
86/86 [==============================] - 4s 46ms/step - loss: 0.6931 - acc: 0.5000 - val_loss: 0.6931 - val_acc: 0.5000
```

```
# Save model
model.save('baseline.keras')
```

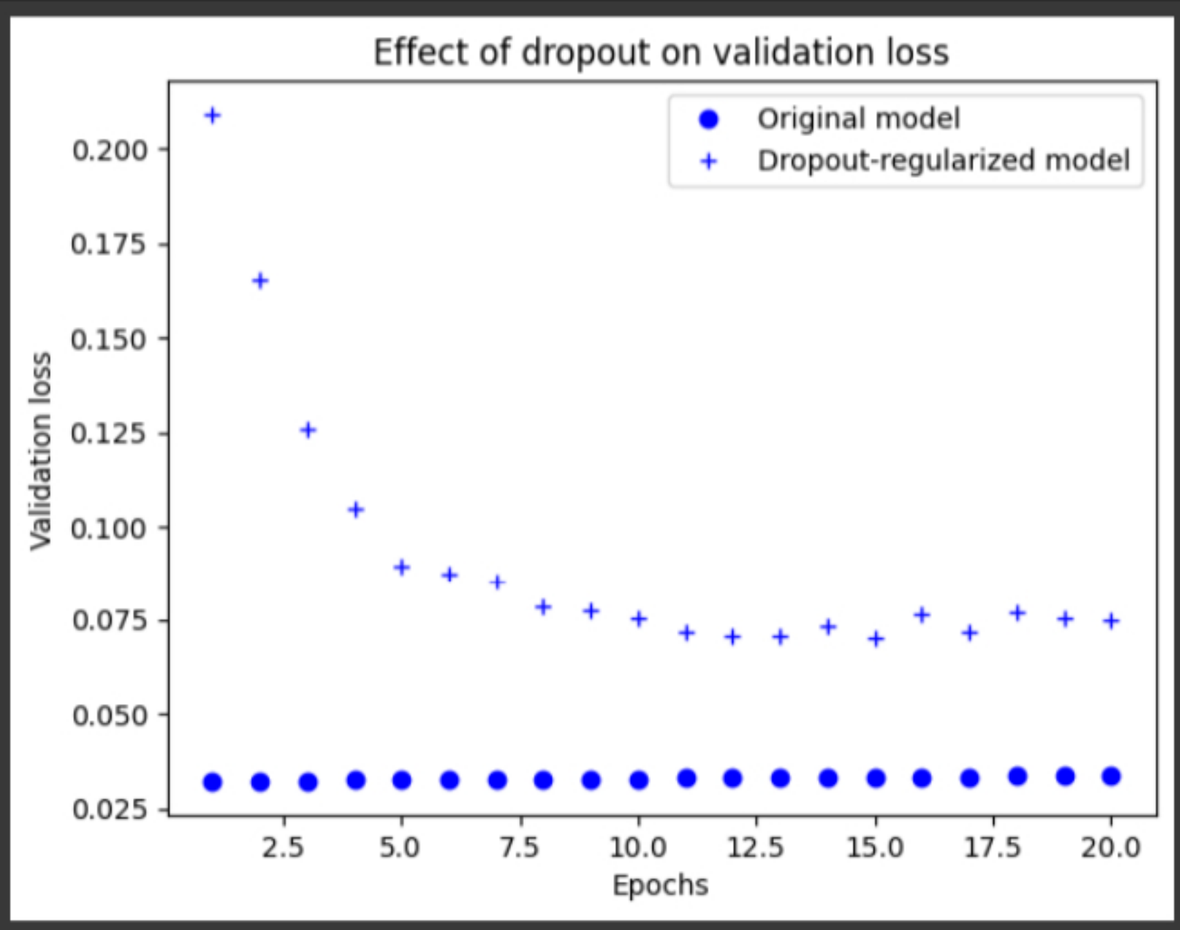
```
# Plot
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], 'bo', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(history.history['acc'], 'bo', label='Training acc')
plt.plot(history.history['val_acc'], 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




###

[88]
```
mod1 = models.Sequential()
mod1.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
mod1.add(layers.Dropout(0.5))
mod1.add(layers.Dense(512, activation='relu'))
mod1.add(layers.Dropout(0.5))
mod1.add(layers.Dense(10, activation='softmax'))
```

[89]
```
compare1(mod, mod1, "dropout", "Dropout-regularized")
```




[90]
```
test(mod1, 5)
```

```
Epoch 1/5
118/118 [==============================] - 5s 32ms/step - loss: 0.0190 - acc: 0.9841
Epoch 2/5
118/118 [==============================] - 6s 48ms/step - loss: 0.0105 - acc: 0.9874
Epoch 3/5
118/118 [==============================] - 4s 31ms/step - loss: 0.0099 - acc: 0.9882
Epoch 4/5
118/118 [==============================] - 4s 33ms/step - loss: 0.0092 - acc: 0.9883
Epoch 5/5
118/118 [==============================] - 4s 35ms/step - loss: 0.0088 - acc: 0.9892
313/313 [==============================] - 1s 4ms/step - loss: 0.0104 - acc: 0.9845
-----
 loss and accuracy:
 [0.010351852513849735, 0.984499990940094]
```

## Summary

### testing data

固定隱藏層為2層，從3.1節可以看到隱藏單元 (hidden units) 為512個時，只需要訓練2次就可以達到最佳模型；從3.2節可以看到隱藏單元分別為64和1024個時，則需要訓練到6~9次才能達到最佳模型。且從下表可發現增加或減少隱藏單元對正確率有些微的提升。

- 2 layers, adjust hidden units

| units | epochs | loss | accuracy | remark |
|---|---|---|---|---|
| 64 | 9 | 0.013 | 0.986 | * smaller |
| 512 | 2 | 0.020 | 0.975 | * base |
| 1024 | 6 | 0.012 | 0.985 | * larger |

固定隱藏層為2層，隱藏單元各512個時，根據3.3節可以發現達到最佳模型所需要的訓練次數增加為4~5次，且從下表可發現加入L2懲罰項所得到的正確率相對另外兩種來的高。

- 2 layers, 512 units, add regularization (alpha=0.001)

| regularization | epochs | loss | accuracy | remark |
|---|---|---|---|---|
| None | 2 | 0.020 | 0.975 | * base |
| L1 | 4 | 0.364 | 0.899 | |
| L2 | 5 | 0.065 | 0.956 | |
| L1 & L2 | 4 | 0.449 | 0.861 | |

固定隱藏層為2層，隱藏單元各512個時，根據3.4節可發現加入了兩層丟棄層 (dropout)，比例皆為0.5時，達到最佳模型所需的訓練次數增加為5次，且從下表可發現增加丟棄層對正確率有些微的提升。

- 2 layers, 512 units, add 2 dropout layers (rate=0.5)

| regularization | epochs | loss | accuracy | remark |
|---|---|---|---|---|
| None | 2 | 0.020 | 0.975 | * base |
| Dropout | 5 | 0.010 | 0.984 | |

### Conclusion

根據上述結果可發現，以隱藏層為2層，隱藏單元各512個之神經網路模型為基礎：3種避免過度配適的方法都會增加達到最佳模型所需要的訓練次數，其中又以改變隱藏單元個數增加的最多次；且3種方法中，除了增加懲罰項這個方法以外，對正確率都有些微的提升。