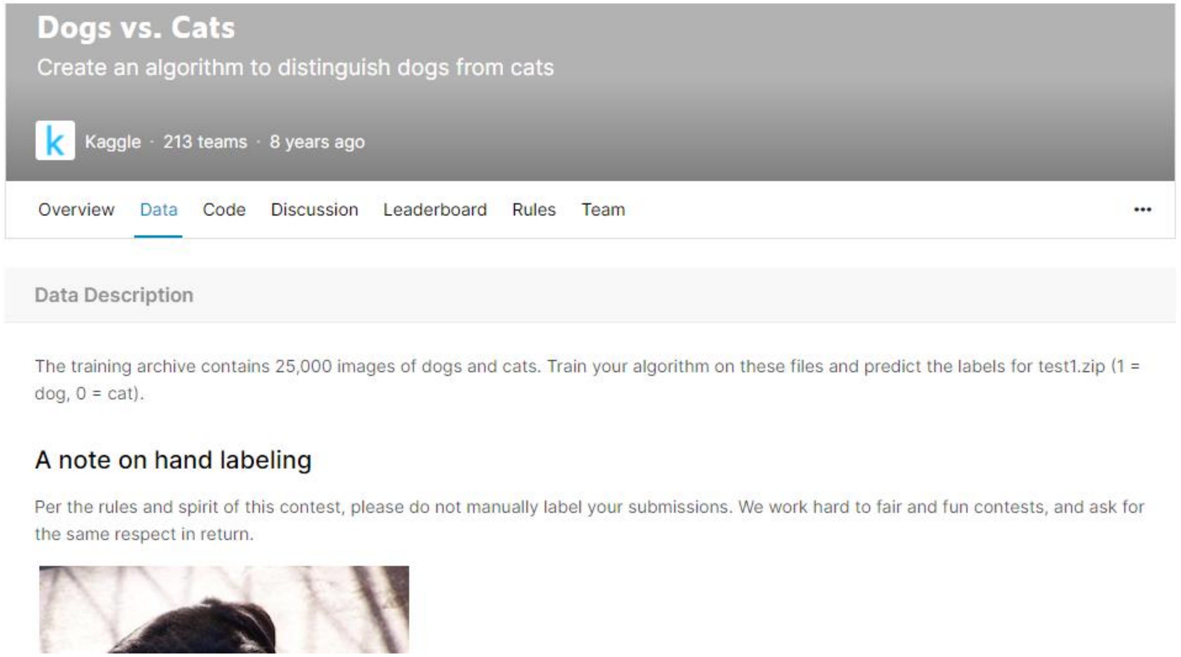
機器學習及其應用

CNN

Data

https://www.kaggle.com/c/dogs-vs-cats/data



Dogs vs. Cats

Create an algorithm to distinguish dogs from cats

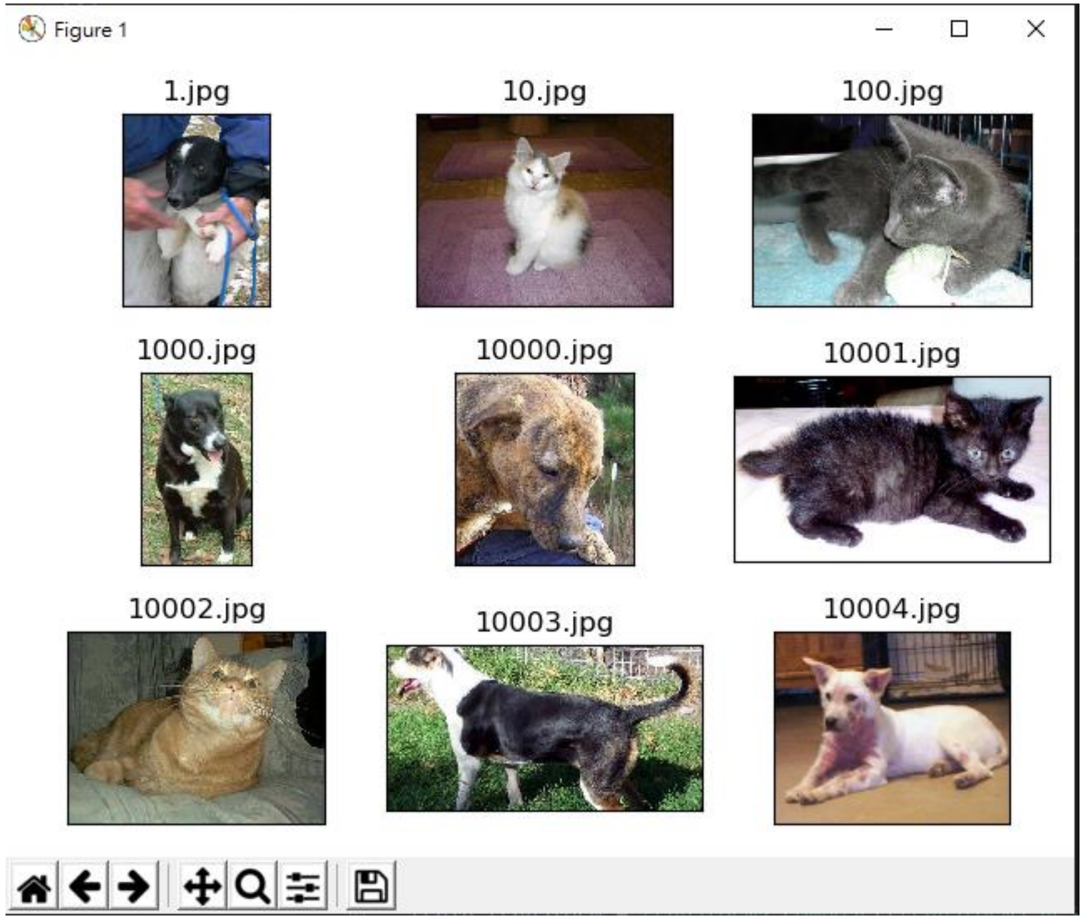
Kaggle · 213 teams · 8 years ago

Data Description

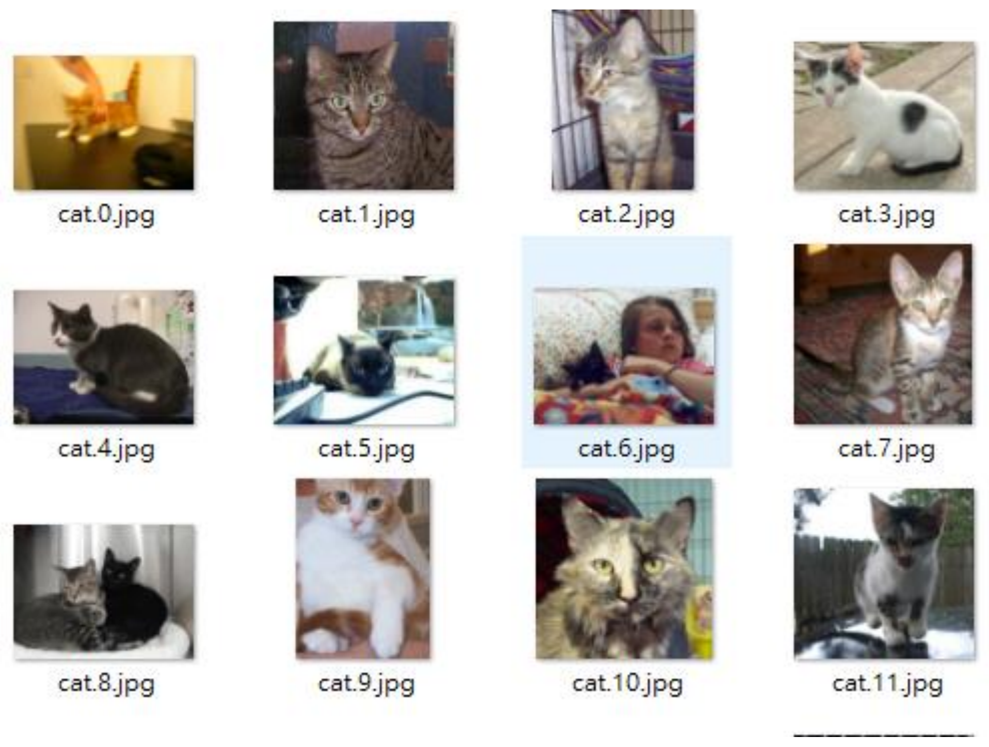
The training archive contains 25,000 images of dogs and cats. Train your algorithm on these files and predict the labels for test1.zip (1 = dog, 0 = cat).

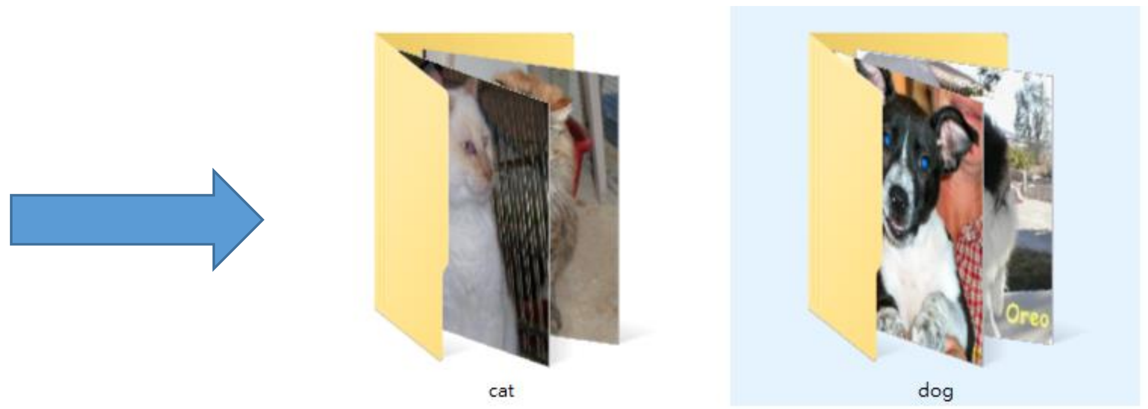
A note on hand labeling

Per the rules and spirit of this contest, please do not manually label your submissions. We work hard to fair and fun contests, and ask for the same respect in return.

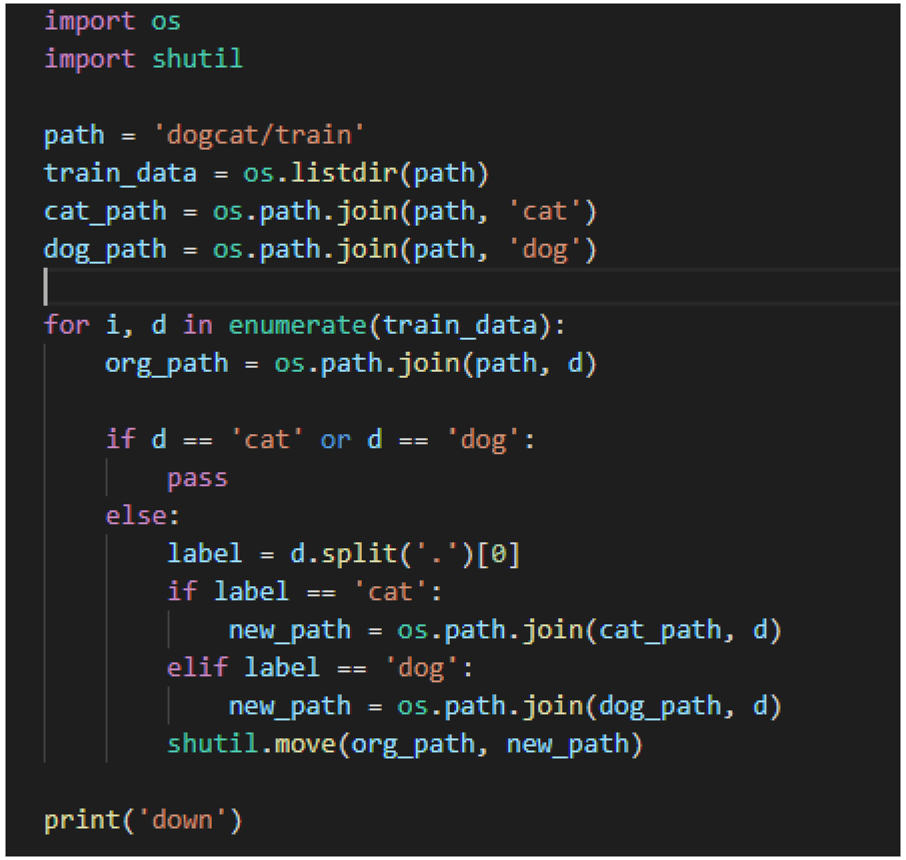


資料處理

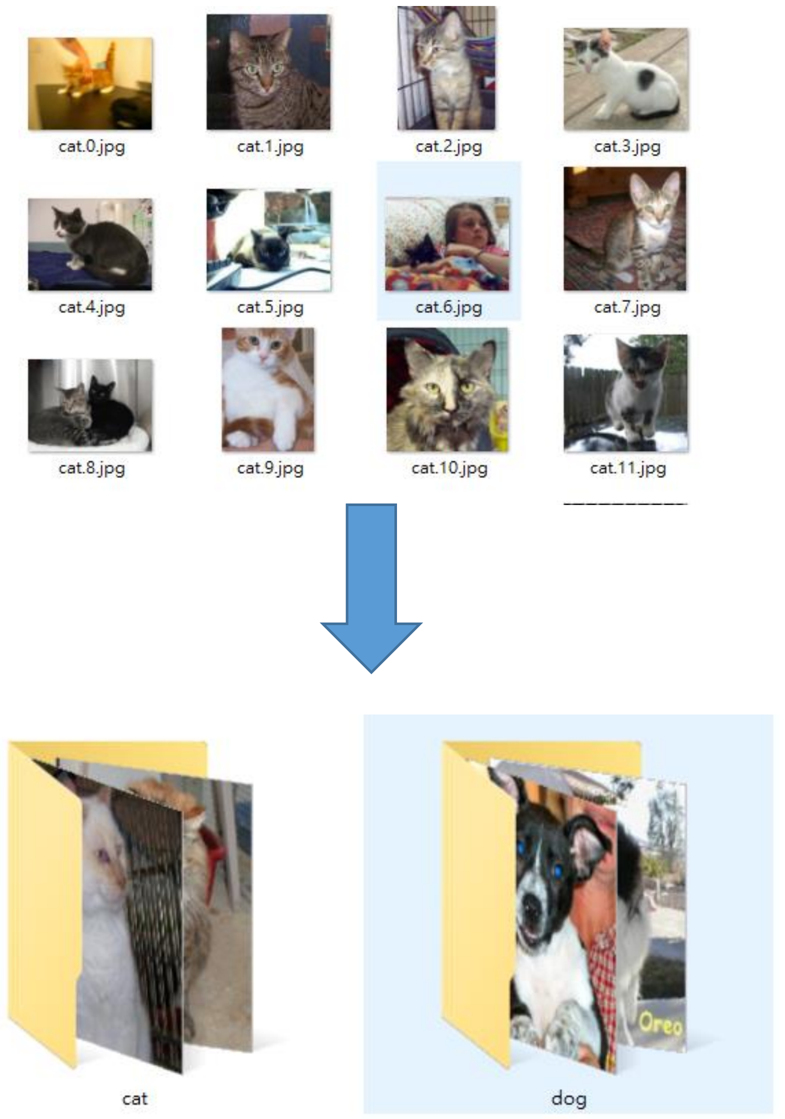

cat.0.jpg
cat.1.jpg
cat.2.jpg
cat.3.jpg
cat.4.jpg
cat.5.jpg
cat.6.jpg
cat.7.jpg
cat.8.jpg
cat.9.jpg
cat.10.jpg
cat.11.jpg


cat
dog

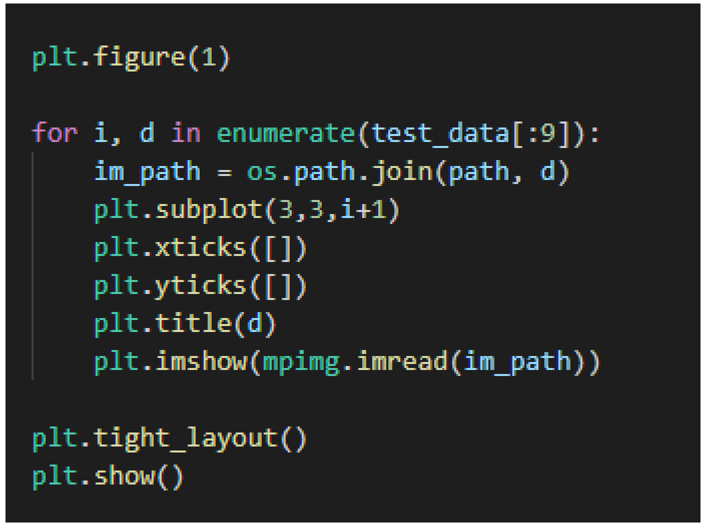
資料處理



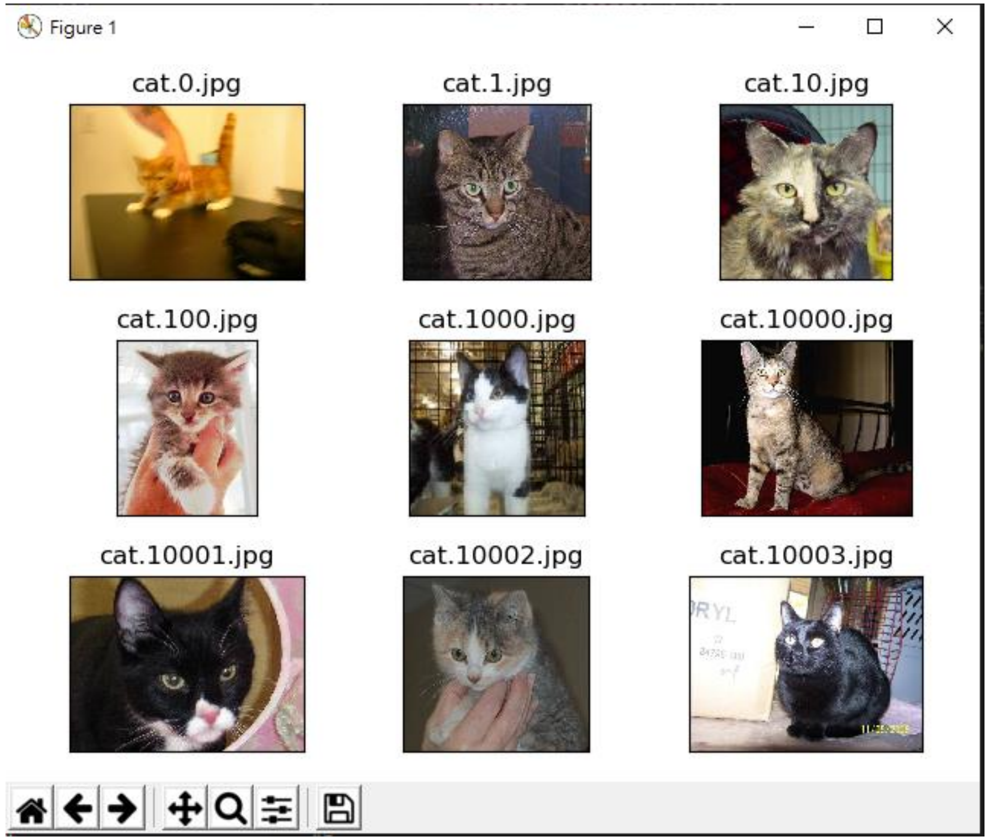
```python
import os
import shutil

path = 'dogcat/train'
train_data = os.listdir(path)
cat_path = os.path.join(path, 'cat')
dog_path = os.path.join(path, 'dog')

for i, d in enumerate(train_data):
    org_path = os.path.join(path, d)

    if d == 'cat' or d == 'dog':
        pass
    else:
        label = d.split('.')[0]
        if label == 'cat':
            new_path = os.path.join(cat_path, d)
        elif label == 'dog':
            new_path = os.path.join(dog_path, d)
        shutil.move(org_path, new_path)

print('down')
```



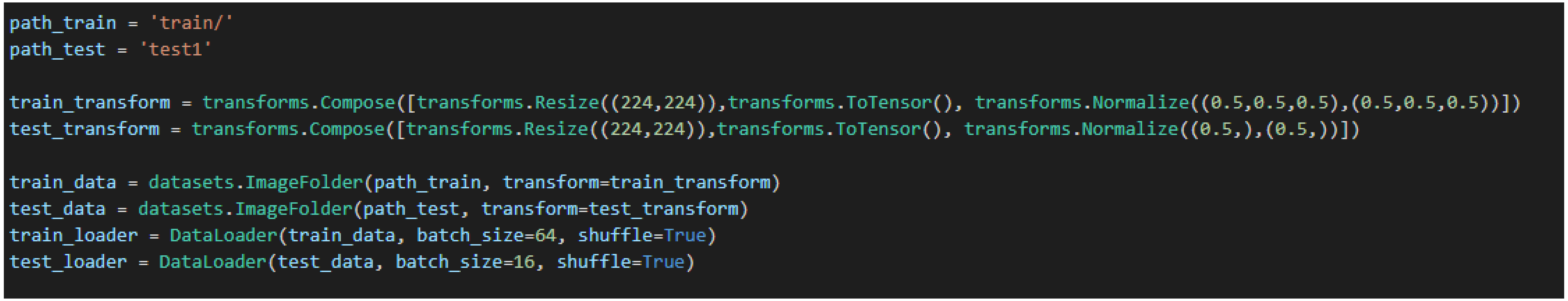
資料處理



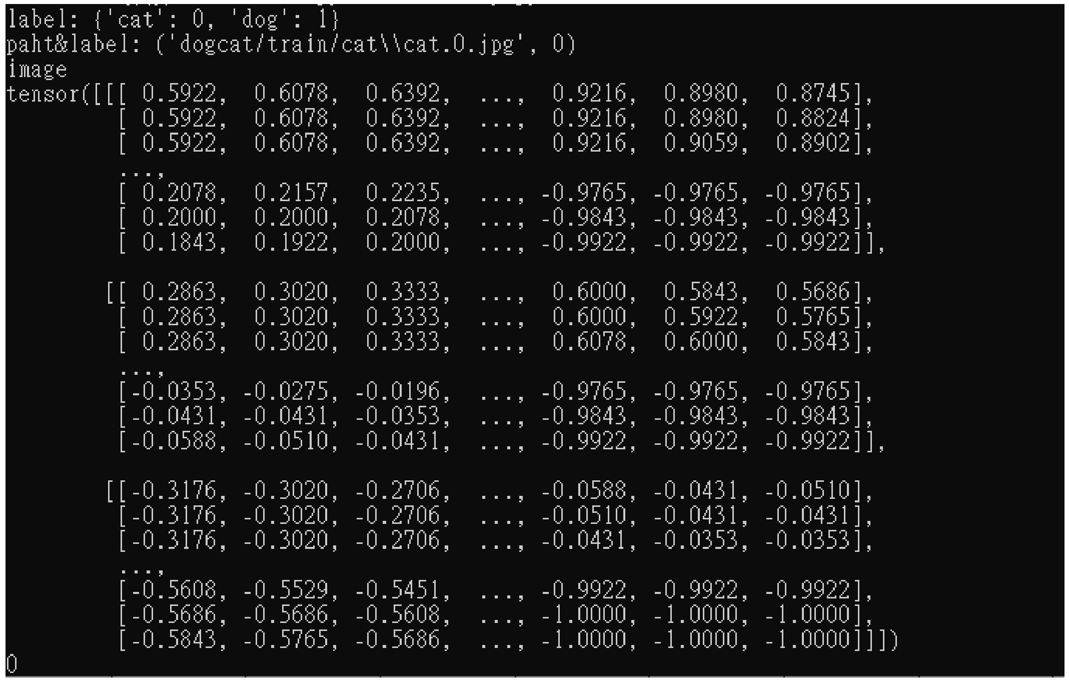
```
plt.figure(1)

for i, d in enumerate(test_data[:9]):
    im_path = os.path.join(path, d)
    plt.subplot(3,3,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.title(d)
    plt.imshow(mpimg.imread(im_path))

plt.tight_layout()
plt.show()
```



資料讀取

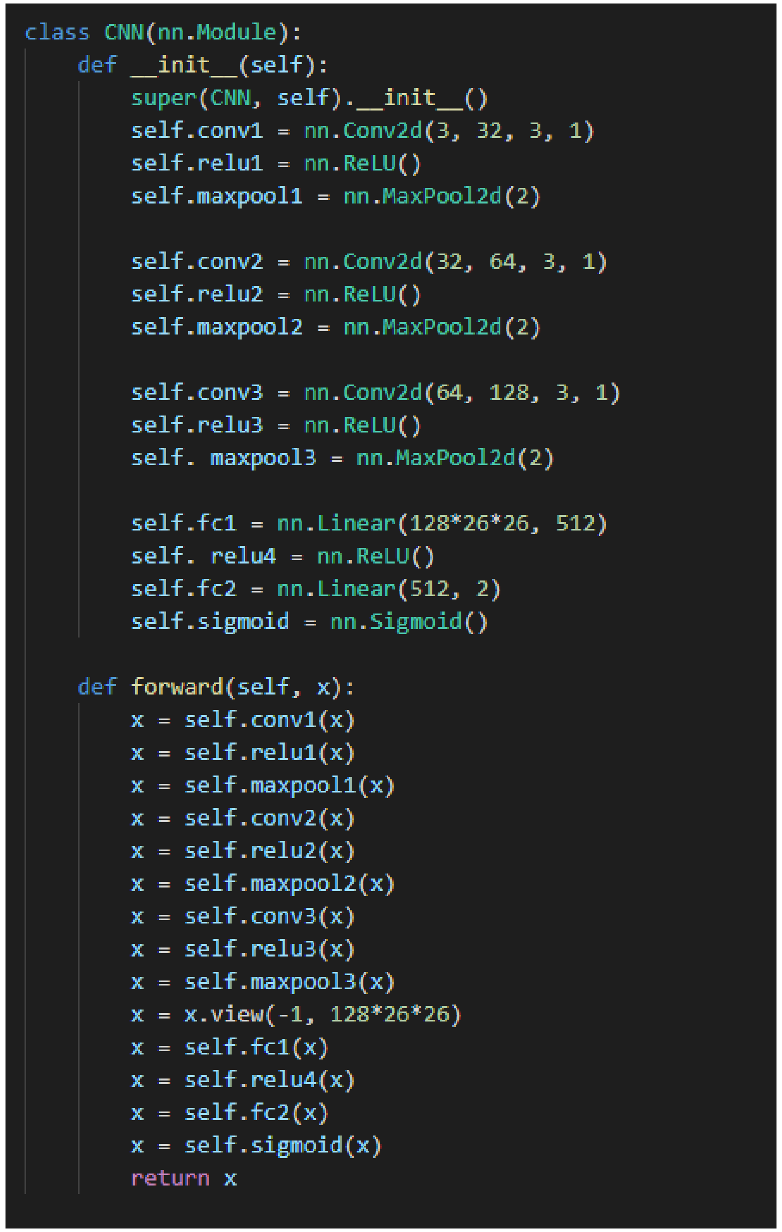


```
path_train = 'train/'
path_test = 'test1'

train_transform = transforms.Compose([transforms.Resize((224,224)),transforms.ToTensor(), transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5))])
test_transform = transforms.Compose([transforms.Resize((224,224)),transforms.ToTensor(), transforms.Normalize((0.5,),(0.5,))])

train_data = datasets.ImageFolder(path_train, transform=train_transform)
test_data = datasets.ImageFolder(path_test, transform=test_transform)
train_loader = DataLoader(train_data, batch_size=64, shuffle=True)
test_loader = DataLoader(test_data, batch_size=16, shuffle=True)
```



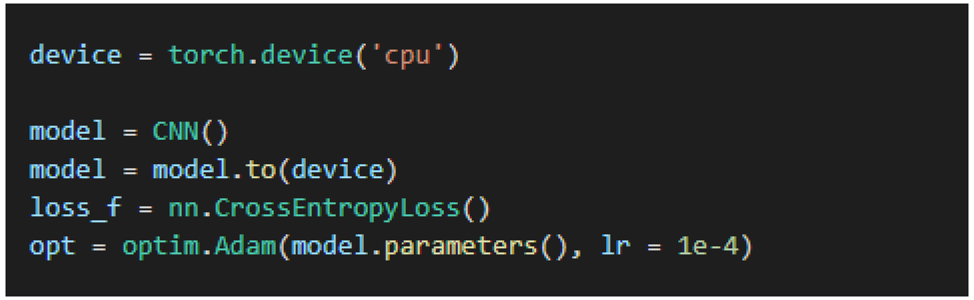
```
label: {'cat': 0, 'dog': 1}
paht&label: ('dogcat/train/cat\\cat.0.jpg', 0)
image
tensor([[[ 0.5922,  0.6078,  0.6392,  ...,  0.9216,  0.8980,  0.8745],
         [ 0.5922,  0.6078,  0.6392,  ...,  0.9216,  0.8980,  0.8824],
         [ 0.5922,  0.6078,  0.6392,  ...,  0.9216,  0.9059,  0.8902],
         ...,
         [ 0.2078,  0.2157,  0.2235,  ..., -0.9765, -0.9765, -0.9765],
         [ 0.2000,  0.2000,  0.2078,  ..., -0.9843, -0.9843, -0.9843],
         [ 0.1843,  0.1922,  0.2000,  ..., -0.9922, -0.9922, -0.9922]],

        [[ 0.2863,  0.3020,  0.3333,  ...,  0.6000,  0.5843,  0.5686],
         [ 0.2863,  0.3020,  0.3333,  ...,  0.6000,  0.5922,  0.5765],
         [ 0.2863,  0.3020,  0.3333,  ...,  0.6078,  0.6000,  0.5843],
         ...,
         [-0.0353, -0.0275, -0.0196,  ..., -0.9765, -0.9765, -0.9765],
         [-0.0431, -0.0431, -0.0353,  ..., -0.9843, -0.9843, -0.9843],
         [-0.0588, -0.0510, -0.0431,  ..., -0.9922, -0.9922, -0.9922]],

        [[-0.3176, -0.3020, -0.2706,  ..., -0.0588, -0.0431, -0.0510],
         [-0.3176, -0.3020, -0.2706,  ..., -0.0510, -0.0431, -0.0431],
         [-0.3176, -0.3020, -0.2706,  ..., -0.0431, -0.0353, -0.0353],
         ...,
         [-0.5608, -0.5529, -0.5451,  ..., -0.9922, -0.9922, -0.9922],
         [-0.5686, -0.5686, -0.5608,  ..., -1.0000, -1.0000, -1.0000],
         [-0.5843, -0.5765, -0.5686,  ..., -1.0000, -1.0000, -1.0000]]])
0
```

model



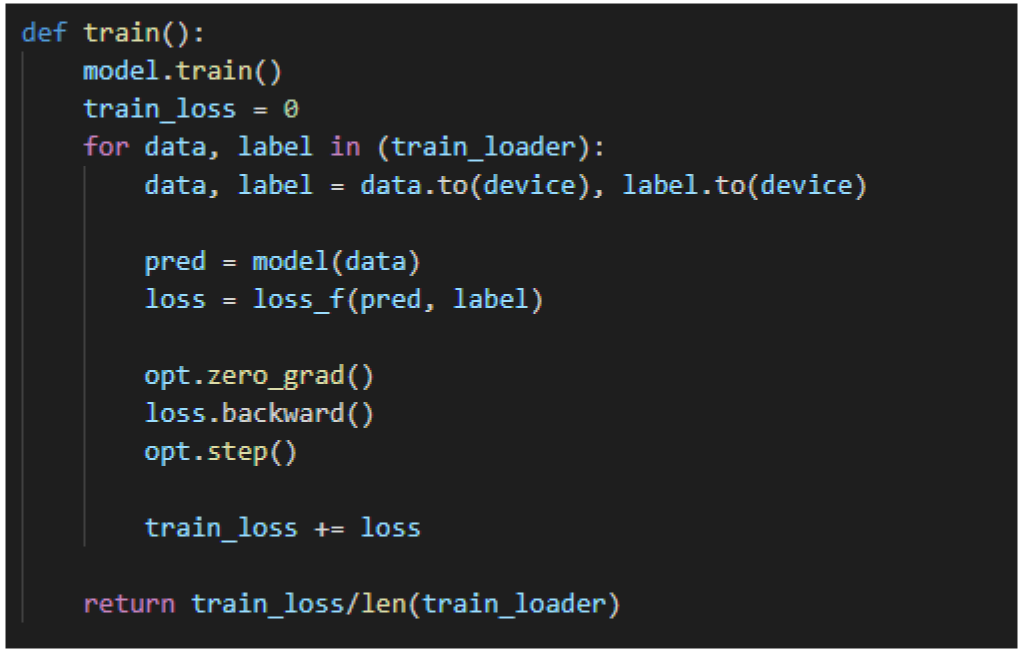
```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, 1)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(2)

        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.relu2 = nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(2)

        self.conv3 = nn.Conv2d(64, 128, 3, 1)
        self.relu3 = nn.ReLU()
        self. maxpool3 = nn.MaxPool2d(2)

        self.fc1 = nn.Linear(128*26*26, 512)
        self. relu4 = nn.ReLU()
        self.fc2 = nn.Linear(512, 2)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.maxpool1(x)
        x = self.conv2(x)
        x = self.relu2(x)
        x = self.maxpool2(x)
        x = self.conv3(x)
        x = self.relu3(x)
        x = self.maxpool3(x)
        x = x.view(-1, 128*26*26)
        x = self.fc1(x)
        x = self.relu4(x)
        x = self.fc2(x)
        x = self.sigmoid(x)
        return x
```

Loss & optim

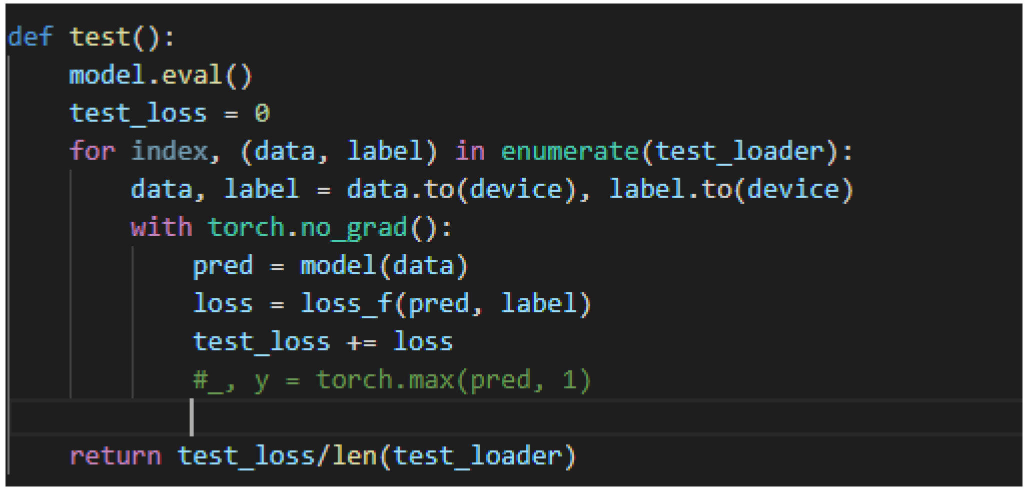


```
device = torch.device('cpu')

model = CNN()
model = model.to(device)
loss_f = nn.CrossEntropyLoss()
opt = optim.Adam(model.parameters(), lr = 1e-4)
```

Train & Test

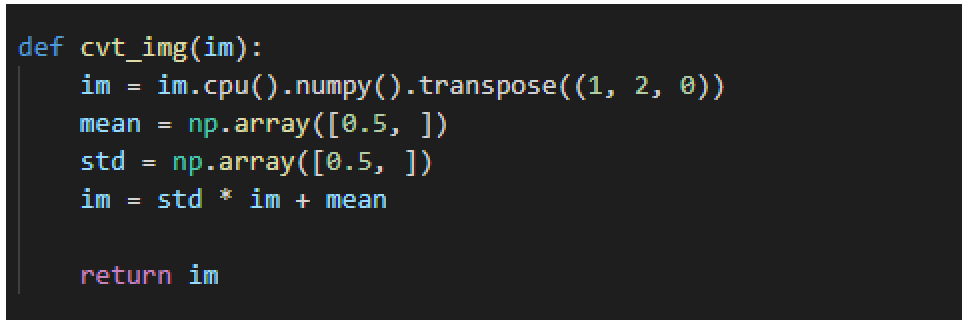


```python
def train():
    model.train()
    train_loss = 0
    for data, label in (train_loader):
        data, label = data.to(device), label.to(device)

        pred = model(data)
        loss = loss_f(pred, label)

        opt.zero_grad()
        loss.backward()
        opt.step()

        train_loss += loss

    return train_loss/len(train_loader)
```

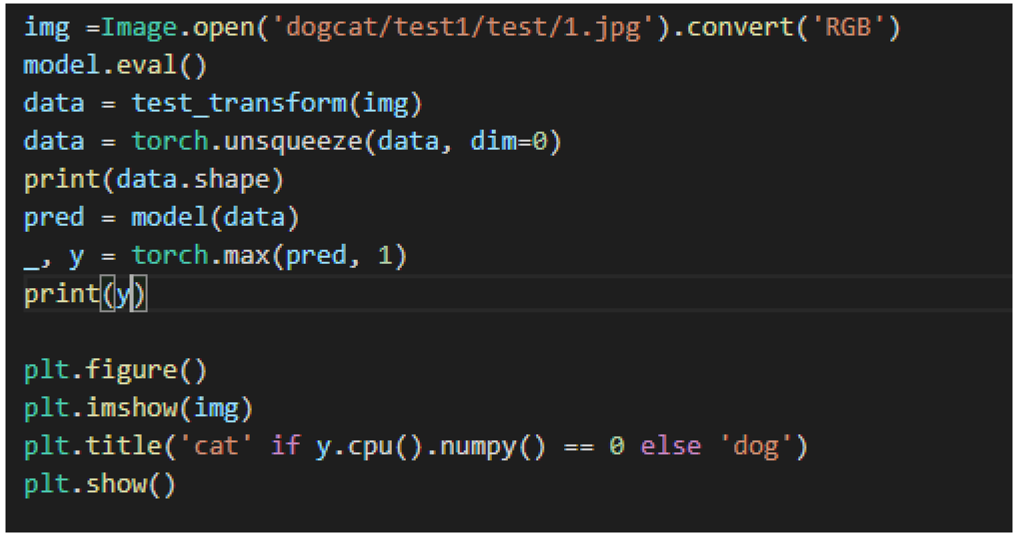


```python
def test():
    model.eval()
    test_loss = 0
    for index, (data, label) in enumerate(test_loader):
        data, label = data.to(device), label.to(device)
        with torch.no_grad():
            pred = model(data)
            loss = loss_f(pred, label)
            test_loss += loss
            #_, y = torch.max(pred, 1)

    return test_loss/len(test_loader)
```

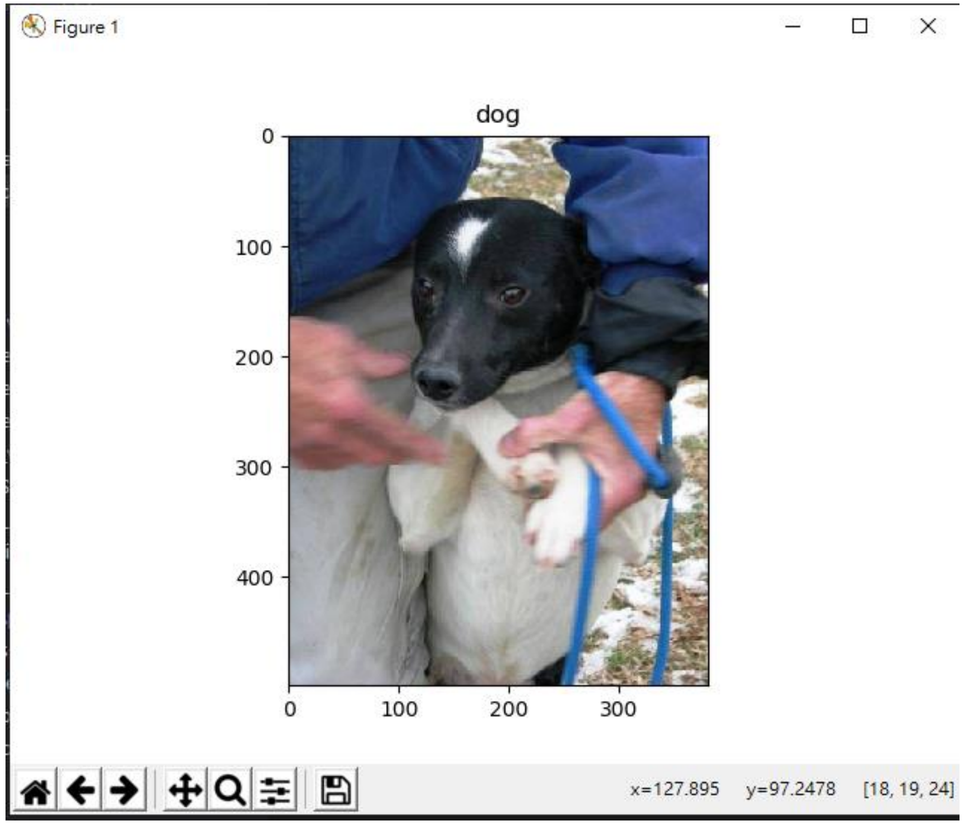
result



```python
def cvt_img(im):
    im = im.cpu().numpy().transpose((1, 2, 0))
    mean = np.array([0.5, ])
    std = np.array([0.5, ])
    im = std * im + mean

    return im
```

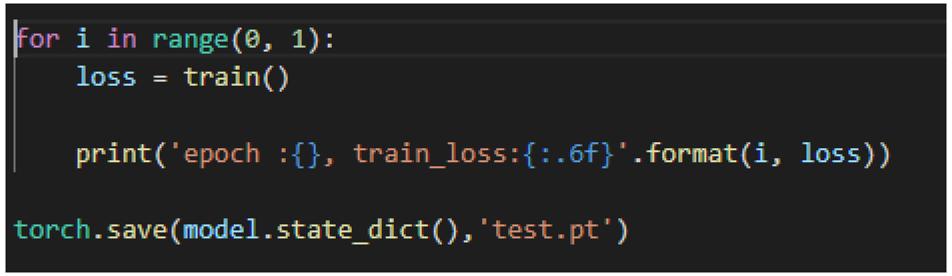


```python
img =Image.open('dogcat/test1/test/1.jpg').convert('RGB')
model.eval()
data = test_transform(img)
data = torch.unsqueeze(data, dim=0)
print(data.shape)
pred = model(data)
_, y = torch.max(pred, 1)
print(y)

plt.figure()
plt.imshow(img)
plt.title('cat' if y.cpu().numpy() == 0 else 'dog')
plt.show()
```



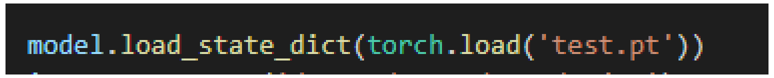
Save model & Load model

Save model



```python
for i in range(0, 1):
    loss = train()

    print('epoch :{}, train_loss:{:.6f}'.format(i, loss))

torch.save(model.state_dict(),'test.pt')
```

Load model



```python
model.load_state_dict(torch.load('test.pt'))
```

期中專案

1. 題目
2. 動機

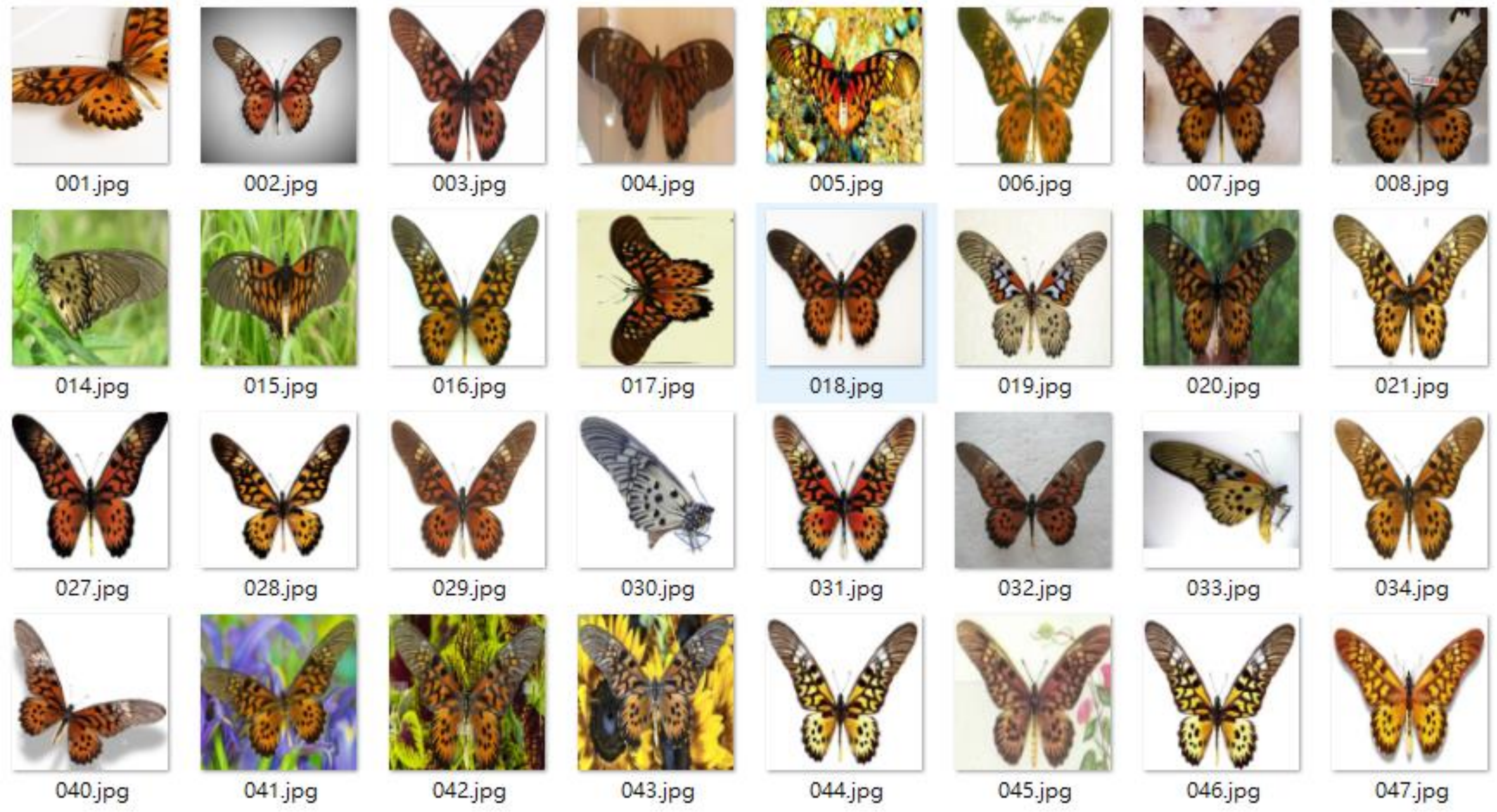
期中報告

1. 動機
2. 方法
3. 程式說明
4. 結果

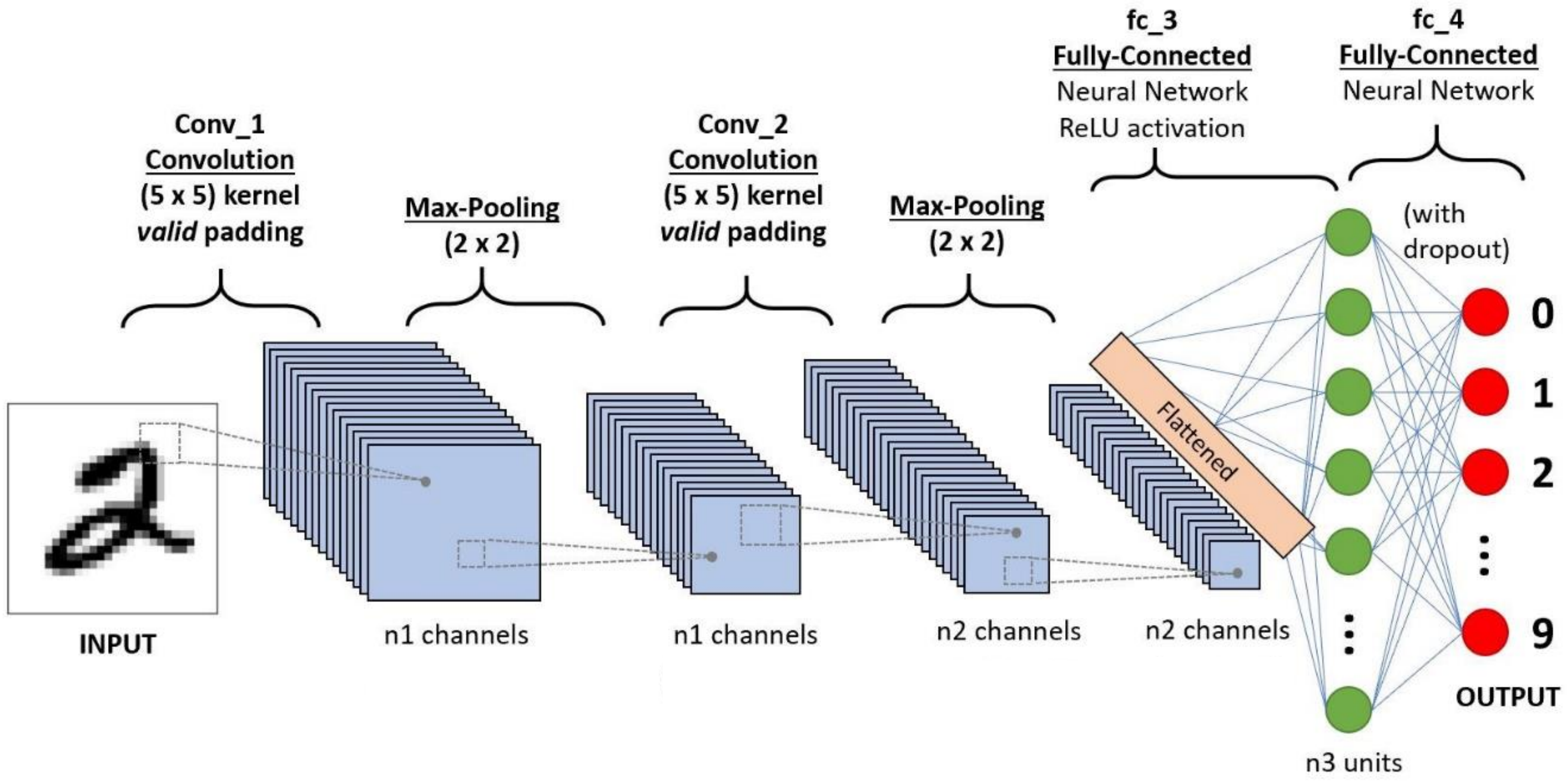
Exercise : butterflies

Data

https://www.kaggle.com/gpiosenka/butterfly-images40-species?select=valid



model



Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
n1 channels
n1 channels
n2 channels
n2 channels
n3 units
0
1
2
9
OUTPUT

result

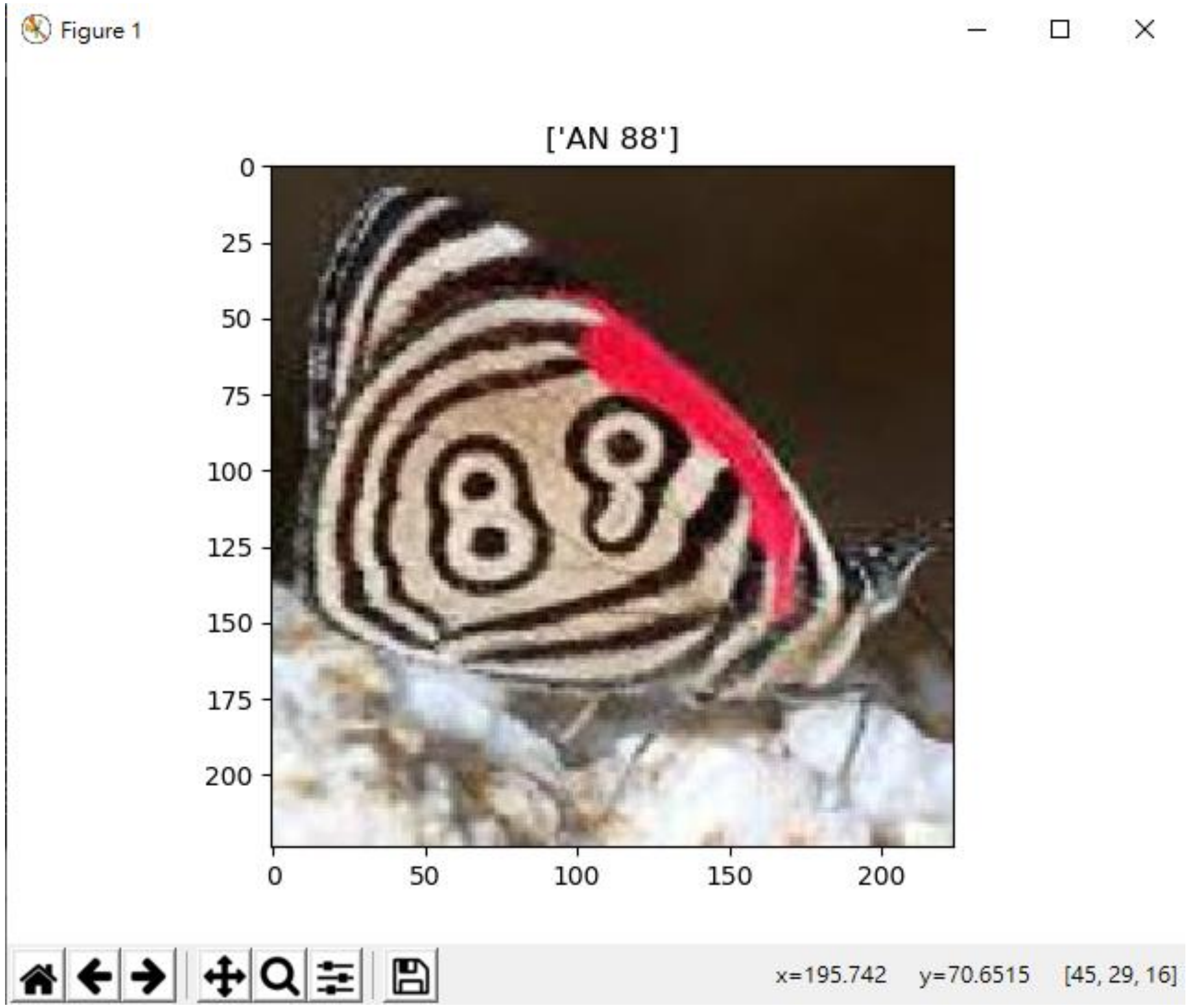


Figure 1
['AN 88']
0
25
50
75
100
125
150
175
200
0
50
100
150
200
x=195.742 y=70.6515 [45, 29, 16]