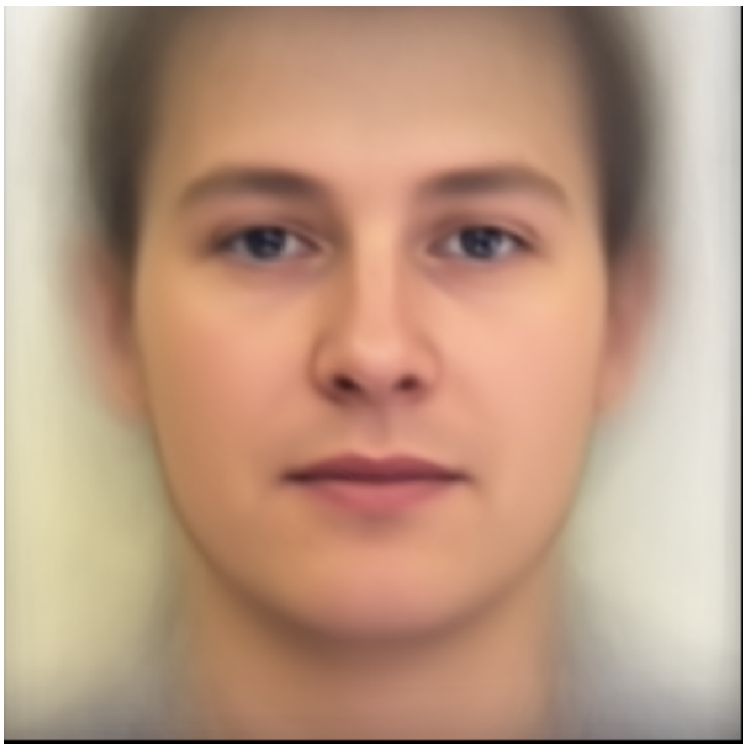
ML Hw4

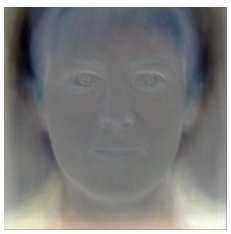
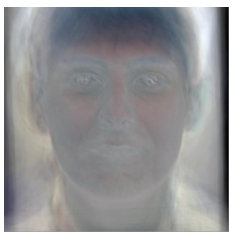
b05902127 資工二 劉俊緯

PCA of colored faces

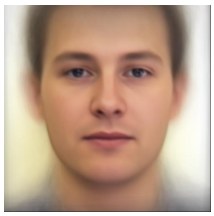
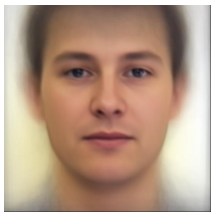
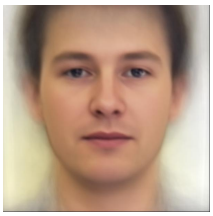
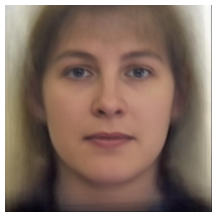
Q1: 所有臉的平均



Q2: 前四Eigenfaces

eigenface	0	1	2	3
picture				

Q3: Reconstruction

reconstruction	12.jpg	27.jpg	127.jpg	217.jpg
picture				

Q4: Proportion

idx	0	1	2	3
%	4.1%	2.9%	2.4%	2.2%

Image clustering

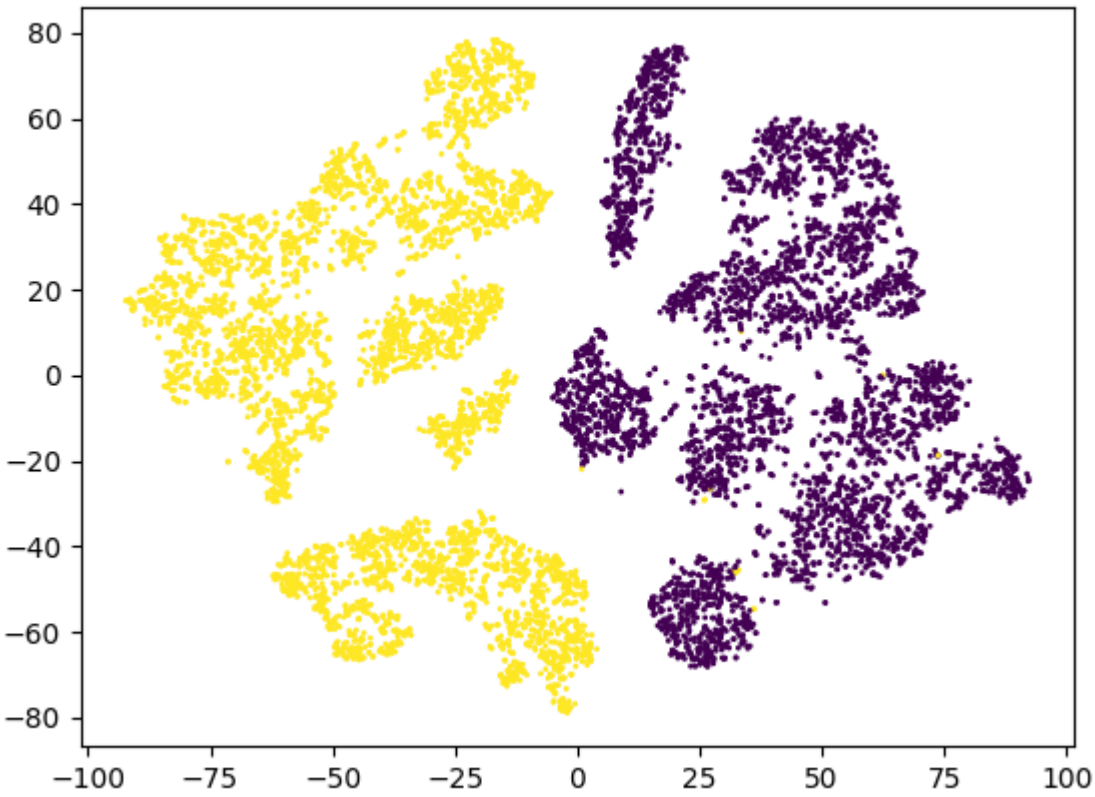
Q1: Two different dim reduction or clustering

method	private score	public score
kmeans	0.51750	0.51674
autoencoder+kmeans	0.99692	0.99683

- 以下image都做過data normalization。
- autoencoder的model。(以下的code + Adam(5e-4) + binary crossentropy)，再用Kmeans降維度。

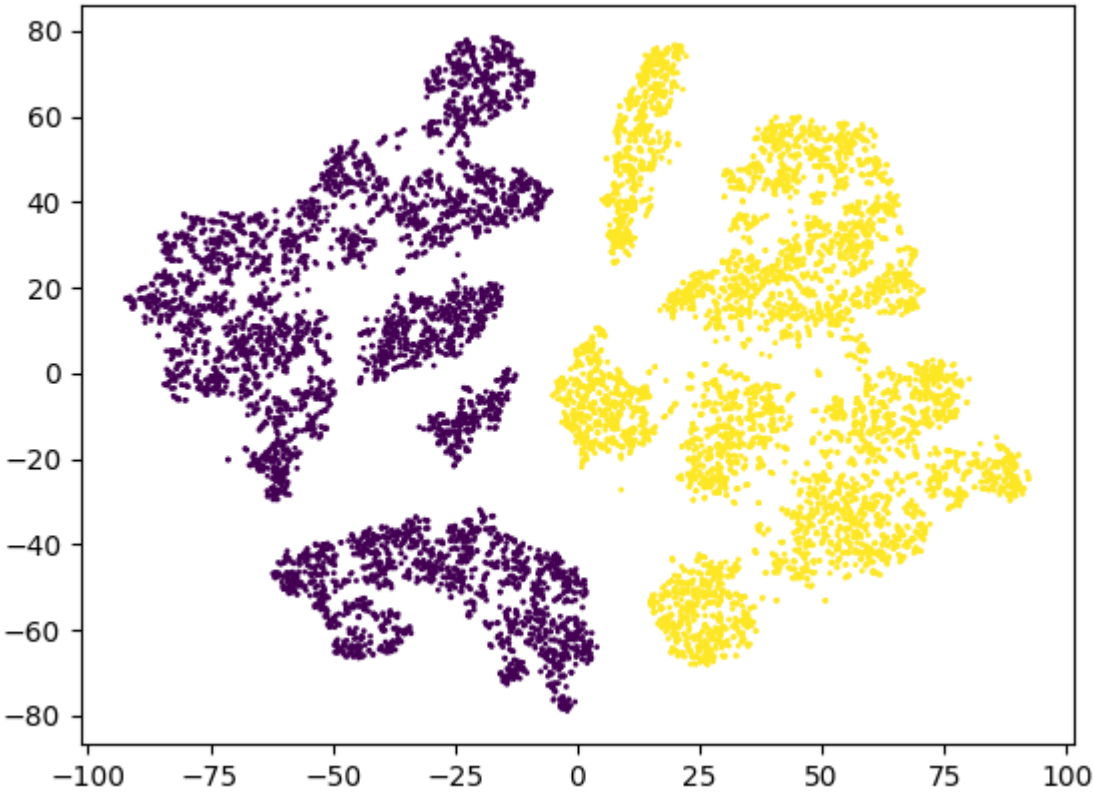
```
x = Input(shape=(28,28,1))
encode = Conv2D(16, (3,3),padding='same')(x)
encode = MaxPooling2D()(encode)
encode = Activation('relu')(encode)
encode = Conv2D(32, (3,3),padding='same')(encode)
encode = MaxPooling2D()(encode)
encode = Activation('relu')(encode)
encode = Conv2D(64, (3,3),padding='same')(encode)
encode = MaxPooling2D()(encode)
encode = Activation('relu')(encode)
encode = Conv2D(128, (3,3),padding='same')(encode)
encode = MaxPooling2D()(encode)
encode = Activation('relu')(encode)
encode = Flatten()(encode)
encode = Dense(64 , activation = 'relu')(encode)
encode = Dense(32 , activation = 'relu')(encode)
decode = Dense(64 , activation = 'relu')(encode)
decode = Dense(128 , activation = 'relu')(encode)
decode = Dense(784 , activation = 'sigmoid')(decode)
decode = Reshape((28,28,1))(decode)
```

Q2: visualization with predict labels



- 使用`sklearn的TSNE(2,init='pca')`視覺化。
- model 是 **Q1**的autoencoder + Kmeans。

Q3: visualization with true labels



- 與原本用Kmeans預測的labels發現，在t-sne降維後的cluster原本距離很近的點，在Kmeans也可能會分錯。

Ensemble learning

Q1: description of my ensemble model

- 將原本的CNN model，分別train 10次，最後用每個對於個別class的probability相加取最大的為投票結果。
- 這10次分別train的model的training data/validation data是使用K-fold作法。
-

model	private score	public score
single model	0.66508	0.67623
single model	0.69434	0.70409

- 單個model的描述：

```
output = BatchNormalization(axis=-1, momentum=0.5)(x)
output = BatchNormalization(axis=-1, momentum=0.5)(conv(32,5,'relu')(output))
#output = Dropout(0.5)(output)
output = BatchNormalization(axis=-1, momentum=0.5)(conv(32,5,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(pool()(output))
output = keras.layers.GaussianNoise(0.1)(output)
output = BatchNormalization(axis=-1, momentum=0.5)(conv(64,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(conv(64,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(pool()(output))

output = BatchNormalization(axis=-1, momentum=0.5)(conv(128,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(conv(128,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(pool()(output))

output = BatchNormalization(axis=-1, momentum=0.5)(conv(256,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(conv(256,3,'relu')(output))
output = BatchNormalization(axis=-1, momentum=0.5)(pool()(output))

output = Flatten()(output)
output = Dropout(0.5)(output)

output = BatchNormalization(axis=-1, momentum=0.5)(den(512)(output))
y = Dense(7, activation='softmax', kernel_initializer='glorot_normal')(output)
ally.append(y)
model[i] = Model(x,y)
### COMPILE ###
model[i].compile(loss='categorical_crossentropy',
                 optimizer='adamax',
                 metrics=['accuracy'])
```