

## 監督式學習 (Supervised Learning)

一開始我就直接拿 Keras 範例 (cifar10\_cnn.py) 的 CNN 架構，含 data augmentation。我只多加旋轉 (順逆時針 10 度) 然後 batch size 設 500。Train 3000 個 epochs，上傳 Kaggle 即有 66.96% 準度！後來再多加幾層 Dense() 並把 activation 改為 LeakyReLU，train 至少 3000 epochs，Kaggle 準度很容易超過 70%！使用的 optimizer 為 Adam，它真的很快！最終架構細節如圖 1 ~ 2。

另外當層數越來越多，每層又越來越大時，dropout 確實有防止 overfitting 效果！多了 dropout 在 train 的時候會發現 training set 精準度下降，但其實在 Kaggle 上不會降，甚至會提昇。

## 半監督式學習 (Semi-Supervised Learning) (1)

第一個半監督方法我用 self-learning：

1. 先用監督式 CNN 架構 train 完 label set。
2. 預測 unlabel set，得到每筆資料每個分類的預測機率。
3. 把分類機率分佈集中者 (entropy 小) 當成分類的預測是正確的，抓進 label set。
  - 把預測結果依機率 (所有分類最高的機率) 排序，發現前 5,000 ~ 10,000 筆機率分佈都不錯。所以就簡單地每次都抓前 5,000 ~ 10,000 筆進來。
  - 前 2,000 筆幾乎沒效果，因為機率分佈又太集中 (entropy 接近 0)。因為把原本就 ~1 的再當成 1 去 train 是不會進步的。
  - 太後面的 entropy 又太高，難以保證預測的正確性。
4. 回到 1. 繼續 train。剩餘的 unlabel set 通常是比較難分類的，加進來 train 可能會誤導，因此剩餘 15,000 筆 unlabel data 時就停止。

通常每次加 unlabel 再 train 的時候，約 200 epochs 內就夠了。

## 半監督式學習 (Semi-Supervised Learning) (2)

第二個方法是 cascaded 架構：先用 autoencoder 抓 features，再用其它 shallow classifier 做分類。

抓 features 我用 Variational AutoEncoder (VAE)，架構也是參考 Keras 範例 XD 細節如圖 3 ~ 5。我把 label 和 unlabel set 都拿去 train。最後把前半 encoder 部份取出 (圖 6)，當成 feature extractor。我取的 feature 維度為 256。

接著用 encoder 抓 label data 的 features，然後用 random forest (圖 7) 當分類器，餵 label data 的 features 和答案給它學。最後再用 encoder 抓 test data features 餵給 random forest 做預測。可是結果很差只有 37%，聽別人說也才約 40% 就不想再試惹 Q\_Q

## 實驗與結果

整體而言準度提昇多是靠修改 CNN 架構 (監督式部份)。使用 self-learning 時，嘗試過各種不同方法，例如一次加多少 unlabel data 進 label set？再 train 時要 train 多少 epochs？最多加到多少 unlabel data？最多約能提昇 3% 準度。

方法	Kaggle 準度
(監督) Keras 範例 CNN，多旋轉	0.66960
(半監督) 架構同上，一次把 unlabel 全加入 label	0.64340
(半監督) 架構同上，每次從 unlabel 加 5000 筆進 label	0.70940
(監督) CNN kernel size=5，多一層 Dense	0.66840
(半監督) 架構同上，每次從 unlabel 加 5000 筆進 label	0.69560
(監督) CNN 比之前再多加 Dense 且 neuron 數增加	0.70740
(半監督) 架構同上，每次從 unlabel 加 5000 筆進 label	0.72340
(監督) 把前面 CNN 的 activation 全改成 LeakyReLU	0.75 以上
(半監督) 架構同上，每次從 unlabel 加 5000 筆進 label	提昇 1% ~ 2%
(半監督) 架構同上，每次從 unlabel 加 10000 筆進 label	提昇 2% ~ 3%

上表是用不同方法在 Kaggle 上的表現。這次作業最大困難點在於 label 資料實在太少，所以很難自己切足夠的 validation set，只好靠上傳看分數惹 Q\_Q

Autoencoder 部份，純粹用它抓的 features 去 train 其它 classifier 效果都不大好。我上傳一次只有 37%，聽其他人也說只有 4x%。因此我覺得 autoencoder 做完後，純用一般 shallow classifier 可能還不夠。

```

104 model = Sequential()
105
106 model.add(Convolution2D(
107     .....64, 3, 3, border_mode='same', input_shape=label_x.shape[1:]))
108 model.add(LeakyReLU(alpha=0.3))
109 model.add(Convolution2D(64, 3, 3))
110 model.add(LeakyReLU(alpha=0.3))
111 model.add(MaxPooling2D(pool_size=(2, 2)))
112 model.add(Dropout(0.25))
113
114 model.add(Convolution2D(32, 3, 3, border_mode='same'))
115 model.add(LeakyReLU(alpha=0.3))
116 model.add(Convolution2D(32, 3, 3))
117 model.add(LeakyReLU(alpha=0.3))
118 model.add(MaxPooling2D(pool_size=(2, 2)))
119 model.add(Dropout(0.25))
120
121 model.add(Flatten())
122 model.add(Dense(512))
123 model.add(LeakyReLU(alpha=0.3))
124 model.add(Dropout(0.5))
125 model.add(Dense(512))
126 model.add(LeakyReLU(alpha=0.3))
127 model.add(Dropout(0.5))
128 model.add(Dense(256))
129 model.add(LeakyReLU(alpha=0.3))
130 model.add(Dropout(0.5))
131 model.add(Dense(128))
132 model.add(LeakyReLU(alpha=0.3))
133 model.add(Dropout(0.25))
134
135 model.add(Dense(10))
136 model.add(Activation('softmax'))
137
138 # Turns out that Adam fucks!
139 # adagrad = Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
140 adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
141 # sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
142 model.compile(
143     .....loss='categorical_crossentropy',
144     .....optimizer=adam,
145     .....metrics=['accuracy'])

```

圖 1. CNN 架構

```

152 # Data augmentation
153 datagen = ImageDataGenerator(
154     .....featurewise_center=False,
155     .....samplewise_center=False,
156     .....featurewise_std_normalization=False,
157     .....samplewise_std_normalization=False,
158     .....zca_whitening=False,
159     .....rotation_range=10,
160     .....width_shift_range=0.10,
161     .....height_shift_range=0.10,
162     .....horizontal_flip=True,
163     .....vertical_flip=False,
164     .....zoom_range=0.15)
165 datagen.fit(label_x)
166
167 model.fit_generator(
168     .....datagen.flow(label_x, label_y, batch_size=500),
169     .....samples_per_epoch=label_x.shape[0],
170     .....nb_epoch=4000,
171     .....validation_data=None)

```

圖 2. Data augmentation

```
116 input = Input(batch_shape=(batch_size,) + (32, 32, 3))
117 conv_1 = Convolution2D(3, 2, 2, border_mode='same', activation='relu')(input)
118 conv_2 = Convolution2D(
119     64, 2, 2,
120     border_mode='same',
121     activation='relu',
122     subsample=(2, 2))(conv_1)
123 conv_3 = Convolution2D(
124     64, 3, 3,
125     border_mode='same',
126     activation='relu',
127     subsample=(1, 1))(conv_2)
128 conv_4 = Convolution2D(
129     64, 3, 3,
130     border_mode='same',
131     activation='relu',
132     subsample=(1, 1))(conv_3)
133 flat = Flatten()(conv_4)
134 hidden = Dense(intermediate_dim, activation='relu')(flat)
135
136 z_mean = Dense(latent_dim)(hidden)
137 z_log_var = Dense(latent_dim)(hidden)
138
139
140 def sampling(args):
141     z_mean, z_log_var = args
142     epsilon = K.random_normal(
143         shape=(batch_size, latent_dim), mean=0.0, std=epsilon_std)
144     return z_mean + K.exp(z_log_var) * epsilon
145
146 #
147 z = Lambda(sampling, output_shape=(latent_dim,))([z_mean, z_log_var])
```

圖 3. VAE 架構 (1)

```

150 decoder_hid = Dense(intermediate_dim, activation='relu')
151 decoder_upsample = Dense(64 * 16 * 16, activation='relu')
152
153 output_shape = (batch_size, 16, 16, 64)
154 decoder_reshape = Reshape(output_shape[1:])
155 decoder_deconv_1 = Deconvolution2D(
156     64, 3, 3,
157     output_shape,
158     border_mode='same',
159     subsample=(1, 1),
160     activation='relu')
161 decoder_deconv_2 = Deconvolution2D(
162     64, 3, 3,
163     output_shape,
164     border_mode='same',
165     subsample=(1, 1),
166     activation='relu')
167
168 output_shape = (batch_size, 33, 33, 64)
169 decoder_deconv_3_upsamp = Deconvolution2D(
170     64, 2, 2,
171     output_shape,
172     border_mode='valid',
173     subsample=(2, 2),
174     activation='relu')
175 decoder_mean_squash = Convolution2D(
176     3, 2, 2,
177     border_mode='valid',
178     activation='sigmoid')
179
180 hid_decoded = decoder_hid(z)
181 up_decoded = decoder_upsample(hid_decoded)
182 reshape_decoded = decoder_reshape(up_decoded)
183 deconv_1_decoded = decoder_deconv_1(reshape_decoded)
184 deconv_2_decoded = decoder_deconv_2(deconv_1_decoded)
185 x_decoded_relu = decoder_deconv_3_upsamp(deconv_2_decoded)
186 x_decoded_mean_squash = decoder_mean_squash(x_decoded_relu)

```

圖 4. VAE 架構 (2)

```

188 def vae_loss(inpu, x_decoded_mean):
189     inpu = K.flatten(inpu)
190     x_decoded_mean = K.flatten(x_decoded_mean)
191     xent_loss = img_rows * img_cols * \
192         objectives.binary_crossentropy(inpu, x_decoded_mean)
193     kl_loss = -0.5 * \
194         K.mean(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
195     return xent_loss + kl_loss
196
197 # Turns out that Adam fucks!
198 # adagrad = Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
199 adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
200 # sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
201 vae = Model(inpu, x_decoded_mean_squash)
202 vae.compile(optimizer=adam, loss=vae_loss)
203 # vae.summary()

```

圖 5. VAE 架構 (3)

```

210 vae.fit(train_x, train_x,
211         shuffle=True,
212         nb_epoch=100,
213         batch_size=batch_size,
214         validation_data=None)
215
216 # Save the 1st model (VAE encoder part).
217 encoder = Model(input, z_mean)
218 encoder.save(out_model + '_1')

```

圖 6. Train 完 VAE 取出其中的 encoder

```

225 #(5000, latent_dim)
226 label_x_encoded = encoder.predict(label_x, batch_size=batch_size, verbose=1)
227
228 #
229 print('Training the classifier...')
230 rf = RandomForestClassifier(n_estimators=300, max_features=128)
231 rf.fit(label_x_encoded, label_y_1.reshape(5000,))

```

圖 7. 用 encoder 抓 features, 再用 random forest 分類