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%！最終架構細節如圖 1 ~ 2。

使用 LeakyReLU 是為避免原 ReLU neuron 輸出 0 時 gradient 也會是 0，即沒任何更新效果。

使用的 optimizer 為 Adam。選它原因是它用 momentum 和針對不同 weights 調整不同 learning rates。Cost function 若有類似狹長山谷形狀，會避免在谷壁來回震盪；又深層網路通常不同 weights 需要不同更新速度，例如靠近 input 的容易有 vanishing gradient problem，可能需要較大 learning rate。比較過 SGD、AdaGrad 和 Adam，發現它速度明顯快很多！

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 (從 92% ~ 93% 降到 88% ~ 89%)，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) 當成分類預測正確，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 100 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 ^^||| 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。

```
model = Sequential()

model.add(Convolution2D(
        64, 3, 3, border_mode='same', input_shape=label_x.shape[1:]))
model.add(LeakyReLU(alpha=0.3))
model.add(Convolution2D(64, 3, 3))
model.add(LeakyReLU(alpha=0.3))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(32, 3, 3, border_mode='same'))
model.add(LeakyReLU(alpha=0.3))
model.add(Convolution2D(32, 3, 3))
model.add(LeakyReLU(alpha=0.3))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(LeakyReLU(alpha=0.3))
model.add(Dropout(0.5))
model.add(Dense(512))
model.add(LeakyReLU(alpha=0.3))
model.add(Dropout(0.5))
model.add(Dense(256))
model.add(LeakyReLU(alpha=0.3))
model.add(Dropout(0.5))
model.add(Dense(128))
model.add(LeakyReLU(alpha=0.3))
model.add(Dropout(0.25))

model.add(Dense(10))
model.add(Activation('softmax'))

# Turns out that Adam fucks!
# adagrad = Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
# sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(
        loss='categorical_crossentropy',
        optimizer=adam,
        metrics=['accuracy'])
```

圖 1. CNN 架構

```
# Data augmentation
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=10,
        width_shift_range=0.10,
        height_shift_range=0.10,
        horizontal_flip=True,
        vertical_flip=False,
        zoom_range=0.15)
datagen.fit(label_x)

model.fit_generator(
        datagen.flow(label_x, label_y, batch_size=500),
        samples_per_epoch=label_x.shape[0],
        nb_epoch=4000,
        validation_data=None)
```

圖 2. Data augmentation

```
inpu = Input(batch_shape=(batch_size,) + (32, 32, 3))
conv_1 = Convolution2D(3, 2, 2, border_mode='same', activation='relu')(inpu)
conv_2 = Convolution2D(
        64, 2, 2,
        border_mode='same',
        activation='relu',
        subsample=(2, 2))(conv_1)
conv_3 = Convolution2D(
        64, 3, 3,
        border_mode='same',
        activation='relu',
        subsample=(1, 1))(conv_2)
conv_4 = Convolution2D(
        64, 3, 3,
        border_mode='same',
        activation='relu',
        subsample=(1, 1))(conv_3)
flat = Flatten()(conv_4)
hidden = Dense(intermediate_dim, activation='relu')(flat)

z_mean = Dense(latent_dim)(hidden)
z_log_var = Dense(latent_dim)(hidden)


def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(
            shape=(batch_size, latent_dim), mean=0.0, std=epsilon_std)
    return z_mean + K.exp(z_log_var) * epsilon

#
z = Lambda(sampling, output_shape=(latent_dim,))([z_mean, z_log_var])
```

圖 3. VAE 架構 (1)

```
decoder_hid = Dense(intermediate_dim, activation='relu')
decoder_upsample = Dense(64 * 16 * 16, activation='relu')

output_shape = (batch_size, 16, 16, 64)
decoder_reshape = Reshape(output_shape[1:])
decoder_deconv_1 = Deconvolution2D(
        64, 3, 3,
        output_shape,
        border_mode='same',
        subsample=(1, 1),
        activation='relu')
decoder_deconv_2 = Deconvolution2D(
        64, 3, 3,
        output_shape,
        border_mode='same',
        subsample=(1, 1),
        activation='relu')

output_shape = (batch_size, 33, 33, 64)
decoder_deconv_3_upsamp = Deconvolution2D(
        64, 2, 2,
        output_shape,
        border_mode='valid',
        subsample=(2, 2),
        activation='relu')
decoder_mean_squash = Convolution2D(
        3, 2, 2,
        border_mode='valid',
        activation='sigmoid')

hid_decoded = decoder_hid(z)
up_decoded = decoder_upsample(hid_decoded)
reshape_decoded = decoder_reshape(up_decoded)
deconv_1_decoded = decoder_deconv_1(reshape_decoded)
deconv_2_decoded = decoder_deconv_2(deconv_1_decoded)
x_decoded_relu = decoder_deconv_3_upsamp(deconv_2_decoded)
x_decoded_mean_squash = decoder_mean_squash(x_decoded_relu)
```

圖 4. VAE 架構 (2)

```
def vae_loss(inpu, x_decoded_mean):
    inpu = K.flatten(inpu)
    x_decoded_mean = K.flatten(x_decoded_mean)
    xent_loss = img_rows * img_cols * \
            objectives.binary_crossentropy(inpu, x_decoded_mean)
    kl_loss = - 0.5 * \
            K.mean(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
    return xent_loss + kl_loss

# Turns out that Adam fucks!
# adagrad = Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
# sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
vae = Model(inpu, x_decoded_mean_squash)
vae.compile(optimizer=adam, loss=vae_loss)
# vae.summary()
```

圖 5. VAE 架構 (3)

```
vae.fit(train_x, train_x,
        shuffle=True,
        nb_epoch=100,
        batch_size=batch_size,
        validation_data=None)

# Save the 1st model (VAE encoder part).
encoder = Model(inpu, z_mean)
encoder.save(out_model + '_1')
```

圖 6. Train 完 VAE 取出其中的 encoder

```
# (5000, latent_dim)
label_x_encoded = encoder.predict(label_x, batch_size=batch_size, verbose=1)

#
print('Training the classifier...')
rf = RandomForestClassifier(n_estimators=300, max_features=128)
rf.fit(label_x_encoded, label_y_1.reshape(5000,))
```

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