國立清華大學

智慧化企業整合
Final Project

基於深度學習網路於手語辨識

指導教授：邱銘傳　教授

110034543　戴佩怡

中　華　民　國　一　一　一　年　一　月　七　日

目錄

圖目錄

表目錄

一、背景介紹

1.1 背景與動機

手語是一個系統的非語言溝通。它通常使用的是聾啞人士。可以使用手語，或者可以以不同的方式來表示。面部表情也可以用來交流思想或想法，為什麼我們應該知道手語。其中一個原因是可以與聾啞人士良好的溝通。如果你不會手語的話，那麼你將不能夠與這些人溝通。

雖然手語在主流社會不普及，但是對聾啞人士族群來說是一個主要語言，也是凝聚這個族群的重要因素。另外，手語就像口語一樣，會因地域關係而有所不同，大則是國家差異，小則有城市之分。手語「方言」很麻煩，作爲一種替代殘缺的語言，不同學校和不同地方的打法都不一樣，學生理解起來困難，很多志工去外地都無法和當地聾啞人溝通。

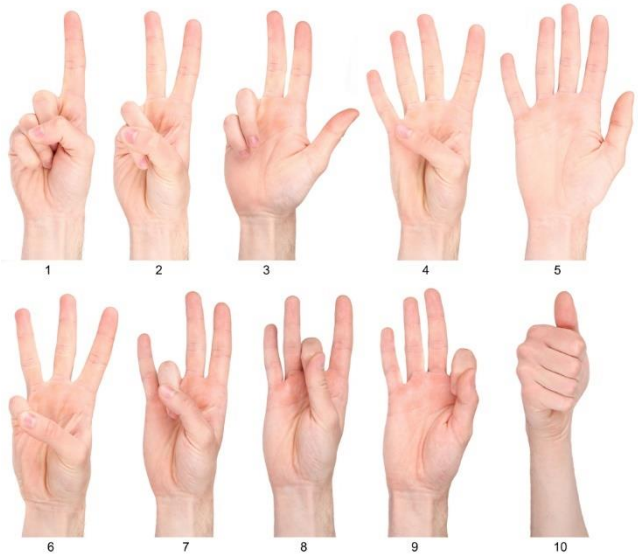


圖 1、手語展示

1.2 研究目的

由於不是每個人都會手語，想藉由手於辨識來幫助聾啞人士與不會手語的族群順利溝通。另外，欲了解深度學習運用於影像辨識之原理並理解如何運用於手語辨識領域，想探討運用 WGAN-GP 來進行資料增強後，是否能提升 CNN 分類模型的準確率。

1.3 問題描述(5W1H)

表 1、5W1H

Why	欲了解深度學習運用於影像辨識之原理並理解如何運用於手語辨識領域
When	欲了解手語表達族群在表達的內容時
Who	需要使用手語表達與溝通的族群
Where	聾啞人士需要表達溝通的任何地方
What	手語準確辨識
How	運用 WGAN-GP 來增加分類模型的訓練以達成準確率的提升

1.4 資料來源

本研究報告使用的資料集來自於 Kaggle 網站中的 American Sign Language Dataset。本資料集總共有 2520 筆手語圖像資料，內容包含數字 0~9 及字母 A~Z 之圖像，此報告採取其中的數字 1~9 類別進行研究實驗。

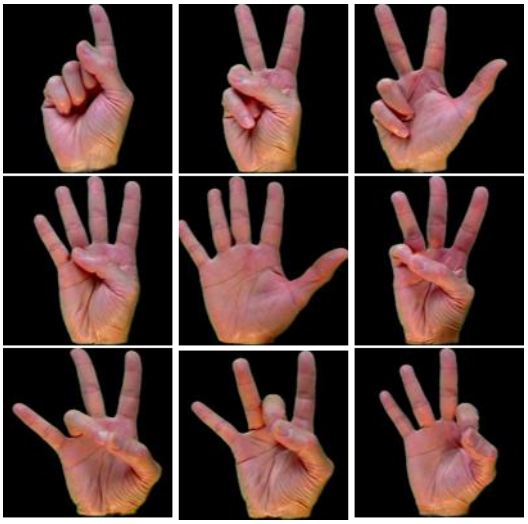


圖 2、1~9 各類別之資料展示

二、研究方法

2.1 方法選擇

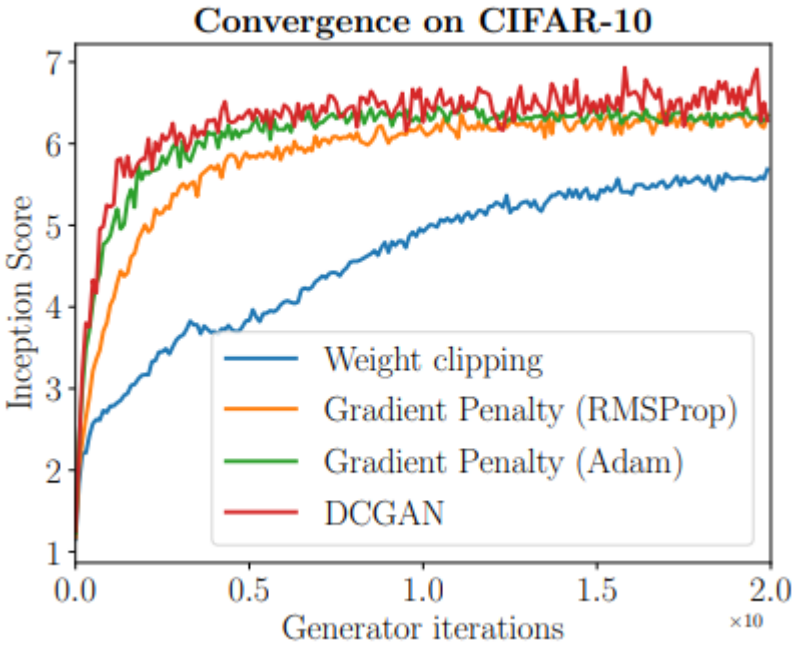


圖 3、各種 GAN 模型之比較

在同樣實驗設置下 WGAN-GP 的結果比 WGAN 效果好，跟 DCGAN 差不多，但是訓練要比 DCGAN 穩定。雖然訓練時間比 DCGAN 要來的長，但因為訓練穩定所以生成的圖片質量要比 DCGAN 好。所以本次研究採 WGAN-GP 來進行實驗。

2.2 研究流程

想透過 WGAN-GP 數據增強方法來達到 CNN 分類模型的準確率提升，再進行超參數的優化，最後分析績效與討論結果，實驗研究流程如圖 4 所示：

圖 4、研究流程

2.3 資料增強

本研究運用圖像旋轉來將原有的 630 筆資料旋轉 90 度、180 度、270 度(如圖 2 所示)，增強至 2520 筆資料。使用之程式碼如圖 3 所示：

圖 5、旋轉資料增強

```
for file in files:
        img = cv2.imread(os.path.join(file_path, file))
        name = file.split(".")
        name_90 = name[0]+"_90.jpg"
        name_180 = name[0]+"_180.jpg"
        name_270 = name[0]+"_270.jpg"

        img90 = np.rot90(img, 1)
        img180 = np.rot90(img, 2)
        img270 = np.rot90(img, 3)

        cv2.imwrite(path+"/"+classs+"/"+name_90, img90)
        cv2.imwrite(path+"/"+classs+"/"+name_180, img180)
        cv2.imwrite(path+"/"+classs+"/"+name_270, img270)
```

圖 6、資料增強程式碼

2.3 WGAN-GP

本研究利用 WGAN-GP（Improved Training of Wasserstein GANs）進行資料增強，其模型結構如圖 3 所示：

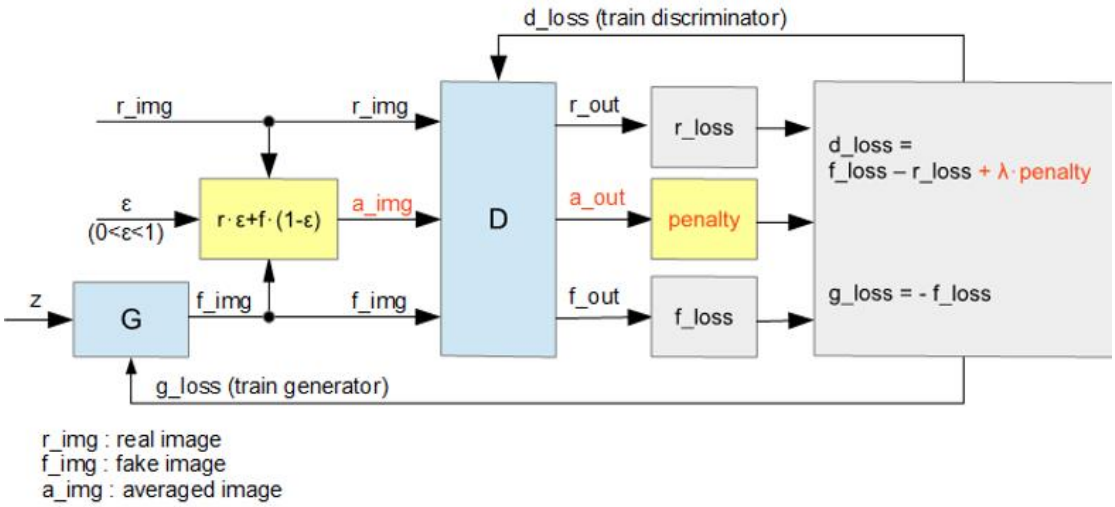


圖 7、WGAN-GP 結構

WGAN-GP 之結構可分為 Generator 與 Discriminator。Generator 之架構設置以圖 8 所示，而 Discriminator 之架構設置以圖 9 所示：

```
class Generator(nn.Module):
    def __init__(self, img_size, latent_dim, dim):
        super(Generator, self).__init__()

        self.dim = dim
        self.latent_dim = latent_dim
        self.img_size = img_size
        self.feature_sizes = (int(self.img_size[0] / 16), int(self.img_size[1] / 16))

        self.latent_to_features = nn.Sequential(
            nn.Linear(latent_dim, 8 * dim * self.feature_sizes[0] * self.feature_sizes[1]),
            nn.ReLU()
        )

        self.features_to_image = nn.Sequential(
            nn.ConvTranspose2d(8 * dim, 4 * dim, 4, 2, 1),
            nn.ReLU(),
            nn.BatchNorm2d(4 * dim),
            nn.ConvTranspose2d(4 * dim, 2 * dim, 4, 2, 1),
            nn.ReLU(),
            nn.BatchNorm2d(2 * dim),
            nn.ConvTranspose2d(2 * dim, dim, 4, 2, 1),
            nn.ReLU(),
            nn.BatchNorm2d(dim),
            nn.ConvTranspose2d(dim, self.img_size[2], 4, 2, 1),
            nn.Sigmoid()
        )
```

圖 8、Generator 之架構設置

```
class Discriminator(nn.Module):
    def __init__(self, img_size, dim):
        """
        img_size : (int, int, int)
            Height and width must be powers of 2.  E.g. (32, 32, 1) or
            (64, 128, 3). Last number indicates number of channels, e.g. 1 for
            grayscale or 3 for RGB
        """
        super(Discriminator, self).__init__()

        self.img_size = img_size

        self.image_to_features = nn.Sequential(
            nn.Conv2d(self.img_size[2], dim, 4, 2, 1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(dim, 2 * dim, 4, 2, 1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(2 * dim, 4 * dim, 4, 2, 1),
            nn.LeakyReLU(0.2),
            nn.Conv2d(4 * dim, 8 * dim, 4, 2, 1),
            nn.Sigmoid()
        )

        # 4 convolutions of stride 2, i.e. halving of size everytime
        # So output size will be 8 * (img_size / 2 ^ 4) * (img_size / 2 ^ 4)
        output_size = int(8 * dim * (img_size[0] / 16) * (img_size[1] / 16))
        self.features_to_prob = nn.Sequential(
            nn.Linear(output_size, 1),
            nn.Sigmoid()
        )
```

圖 9、Discriminator 之架構設置

2.4 VGG19

本研究使用之深度學習模型為 VGG19，由 16 個卷積層與 3 個全連接層以及 5 個池化層（Pool layer）組成，分別用 maxpool 表示，其模型架構如圖 10 所示：

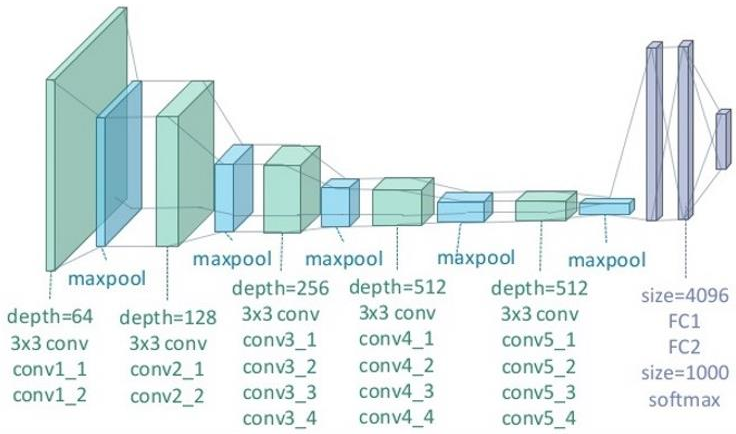


圖 10、VGG19 架構

VGG19 之架構設置如圖 11 至圖 12 所示：

Model: "vgg19"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 48, 48, 3)]	0
block1_conv1 (Conv2D)	(None, 48, 48, 64)	1792
block1_conv2 (Conv2D)	(None, 48, 48, 64)	36928
block1_pool (MaxPooling2D)	(None, 24, 24, 64)	0
block2_conv1 (Conv2D)	(None, 24, 24, 128)	73856
block2_conv2 (Conv2D)	(None, 24, 24, 128)	147584
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv4 (Conv2D)	(None, 12, 12, 256)	590080
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
block4_conv1 (Conv2D)	(None, 6, 6, 512)	1180160
block4_conv2 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv3 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv4 (Conv2D)	(None, 6, 6, 512)	2359808
block4_pool (MaxPooling2D)	(None, 3, 3, 512)	0
block5_conv1 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv2 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv3 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv4 (Conv2D)	(None, 3, 3, 512)	2359808
block5_pool (MaxPooling2D)	(None, 1, 1, 512)	0

圖 11、VGG19 架構設置 1

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg19 (Functional)	(None, 1, 1, 512)	20024384
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 1024)	525312
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 9)	9225

Total params: 20,558,921
Trainable params: 20,558,921
Non-trainable params: 0

圖 12、VGG19 架構設置 2

使用 ReduceLROnPlateau 之情形為當模型連續訓練 n 次準確率沒有提升，就自動進行學習率衰降的調整，可以在訓練過程中優化學習率，閾值最下限為 1e-10。

如圖 13 所示：

```
lrd = ReduceLROnPlateau(monitor = 'val_loss',patience = 20
                        ,verbose = 1,factor = 0.50, min_lr = 1e-10)
```

圖 13、VGG16 之學習率設置

三、實驗過程

3.1 資料處理

資料處裡的部分則是將原資料集進旋轉增強後行資料切割，分為訓練集、驗證集，測試集的部分則是自行蒐集，各資料集筆數之比例為 80%、10%、10%。

WGAN-GP 再將分割好的訓練資料集進行一倍數的資料增強作為另一組訓練集(如圖 15、表 3 所示)。三資料集之圖片數量處理以表 2 所示：

表 2、三資料集比例與數量

WGAN-GP	Train set	Val set	Test set
No	252	28	28
Yes	504	28	28

Test set 資料來源為自己收集之資料，取 14 人所比之 1~9 數字手語，各類別每個人拍兩張，總共收集到 28 筆資料。

收集資料樣本如圖 14 所示：

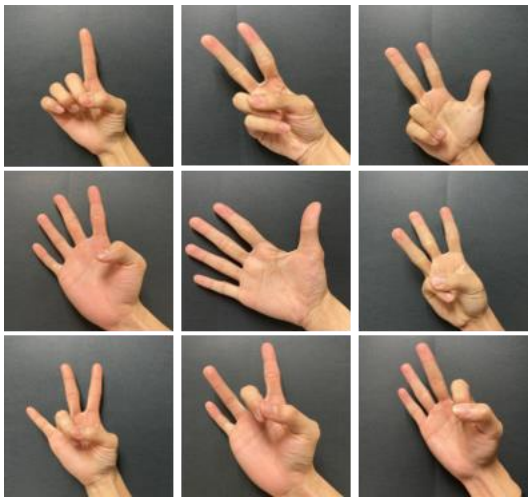


圖 14、Test 集樣本

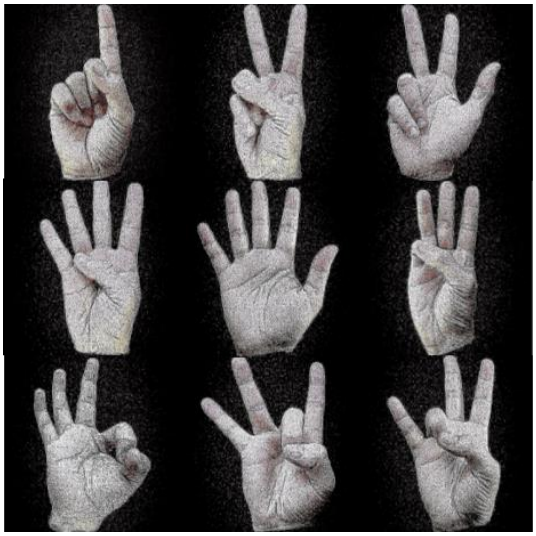
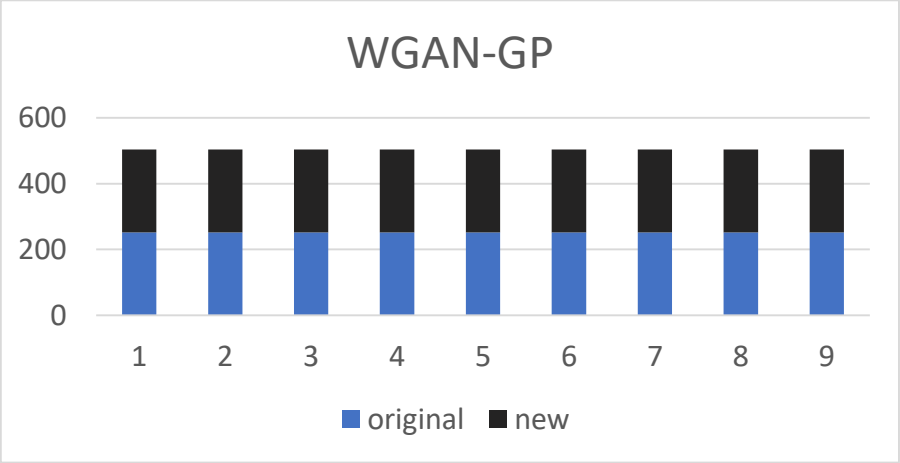


圖 15、WGAN-GP 生成之資料

表 3、資料增強後訓練數量



3.2 分類結果

首先進行無 WGAN-GP 資料增強之訓練，VGG19 初始參數設定如表 4 所示。訓練結果如圖 16 所示，測試結果表 5 所示。

表 4、初始參數設定

參數	設定
Optimizer	adam
Batch size	32
Dropout	0.5
Epoch	30

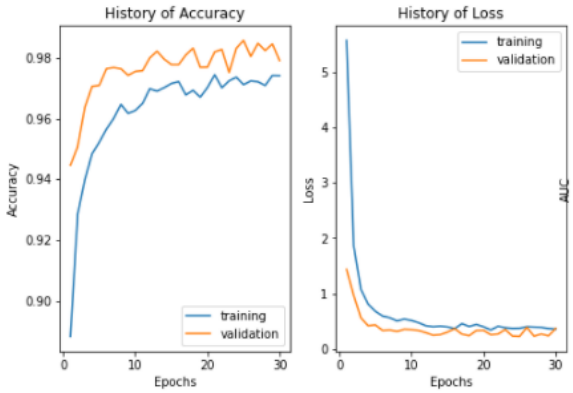


圖 16、初始訓練結果

表 5、初始測試集準確率

	Accuracy
Train	0.9741
Val	0.9862
Test	0.8466

再來將 WGAN-GP 增強之訓練資料讓 VGG19 訓練，訓練參數與上述初始參數設定相同，訓練結果如圖 17 所示，測試結果如表 6 所示。

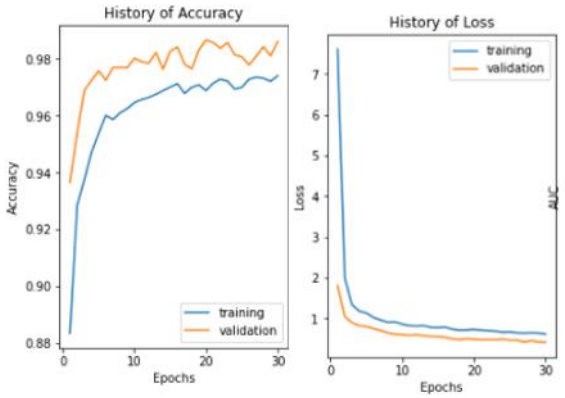


圖 17、資料增強訓練結果

	Accuracy
Train	0.9744
Val	0.9862
Test	0.8758

可以觀察到，透過 WGAN-GP 資料增強後的資料確實提升了測試集的準確率，訓練過程也很穩定。

四、參數優化

4.1 實驗設計

本實驗利用田口方法的實驗設計，使用直交表以較少的實驗來獲得更可靠的因子效果估計。進行 3 因子 3 水準之實驗設計，其中選定之因子為 Optimizer、Dropout、Batch Size。實驗設計之表格如表 5 所示：

表 6、實驗設計表格

Factor / Level	Optimizer	Dropout	Batch Size
Level 1	Adam	0.4	16
Level 2	AdaDelta	0.5	32
Level 3	AdaGrad	0.6	64

4.2 L_9 直交表

實驗設計之結果以表 6 之 L_9 直交表所示：

表 7、L9 直交表

Factor / Experimant	Optimizer	Dropout	Batch Size	Accuracy
1	Adam	0.4	16	0.8532
2	Adam	0.5	32	0.8658
3	Adam	0.6	64	0.8576
4	AdaDelta	0.5	64	0.8247
5	AdaDelta	0.6	16	0.8463
6	AdaDelta	0.4	32	0.8132
7	AdaGrad	0.6	32	0.8465
8	AdaGrad	0.4	64	0.8532
9	AdaGrad	0.5	16	0.8469

4.3Minitab 結果

由主效果分析結果可以看出，最佳參數組合為 Optimizer：Adam、Dropout：0.5、Batch Size：64。

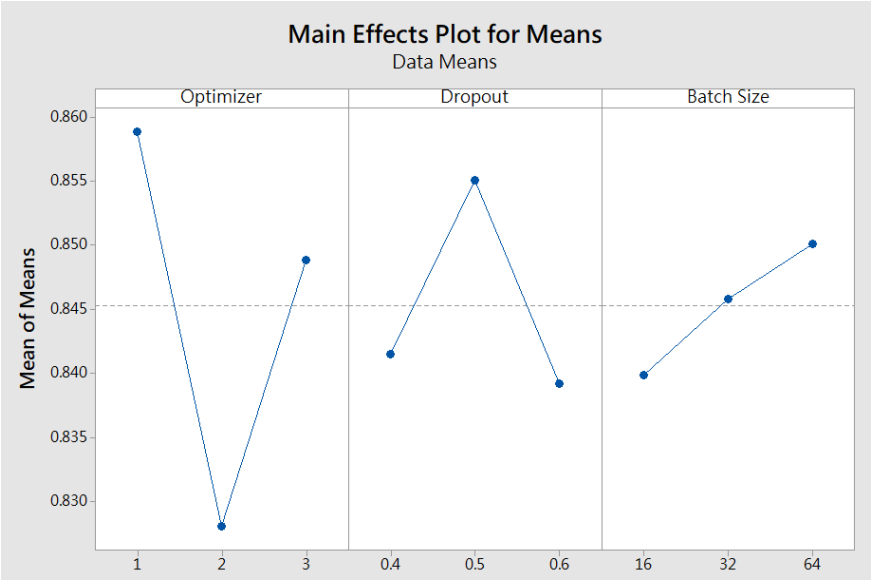


圖 18、主效果分析

Response Table for Means

Level	Optimizer	Dropout	Batch Size
1	0.8589	0.8415	0.8399
2	0.8281	0.8551	0.8458
3	0.8489	0.8392	0.8501
Delta	0.0308	0.0159	0.0103
Rank	1	2	3

圖 19、主效果排名

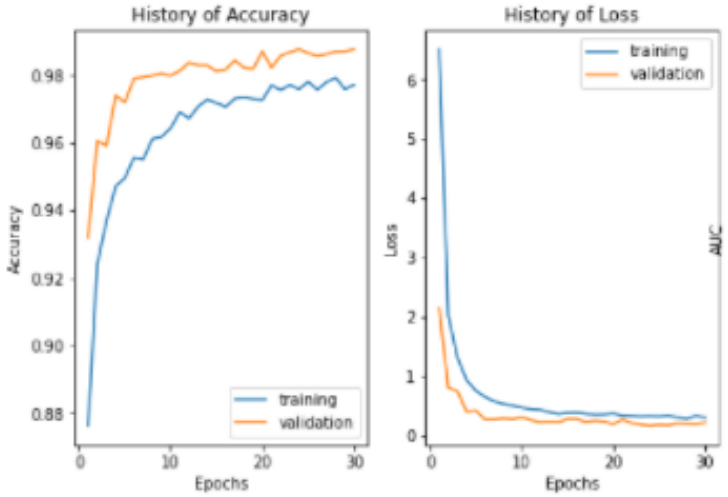


圖 20、最佳組合之訓練結果

表 8、最佳組合測試結果

	Accuracy
Train	0.9846
Val	0.9873
Test	0.8689

由主因子分析所分析之最佳組合的準確率確實為最佳結果。

五、結論

5.1 研究結果與未來展望

- 透過 WGAN-GP 數據增強後的資料，進行 VGG19 訓練，結果提高手語分類的準確率。
- 利用自行收集的 Test 資料集，驗證所訓練模型的泛化能力。
- 除了手語的辨識外，未來可將手語辨識結合翻譯機功能，讓不會手語的族群也能方便快速的與手與族群順利溝通。

六、參考文獻

[1] **Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans.** ***arXiv preprint arXiv:1704.00028*****.**

[2] **Gao, X., Deng, F., & Yue, X. (2020). Data augmentation in fault diagnosis based on the Wasserstein generative adversarial network with gradient penalty.** ***Neurocomputing, 396,*** **487-494.**

[3] **https://cloud.tencent.com/developer/article/1645877**

[4] **http://www.twistedwg.com/2018/02/02/WGAN-GP.html**

[5] **https://zhuanlan.zhihu.com/p/52799555**