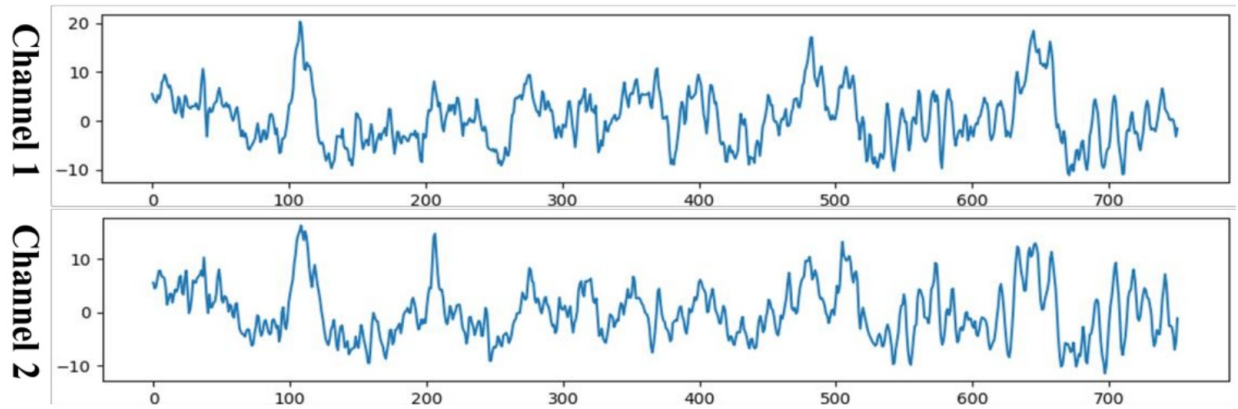
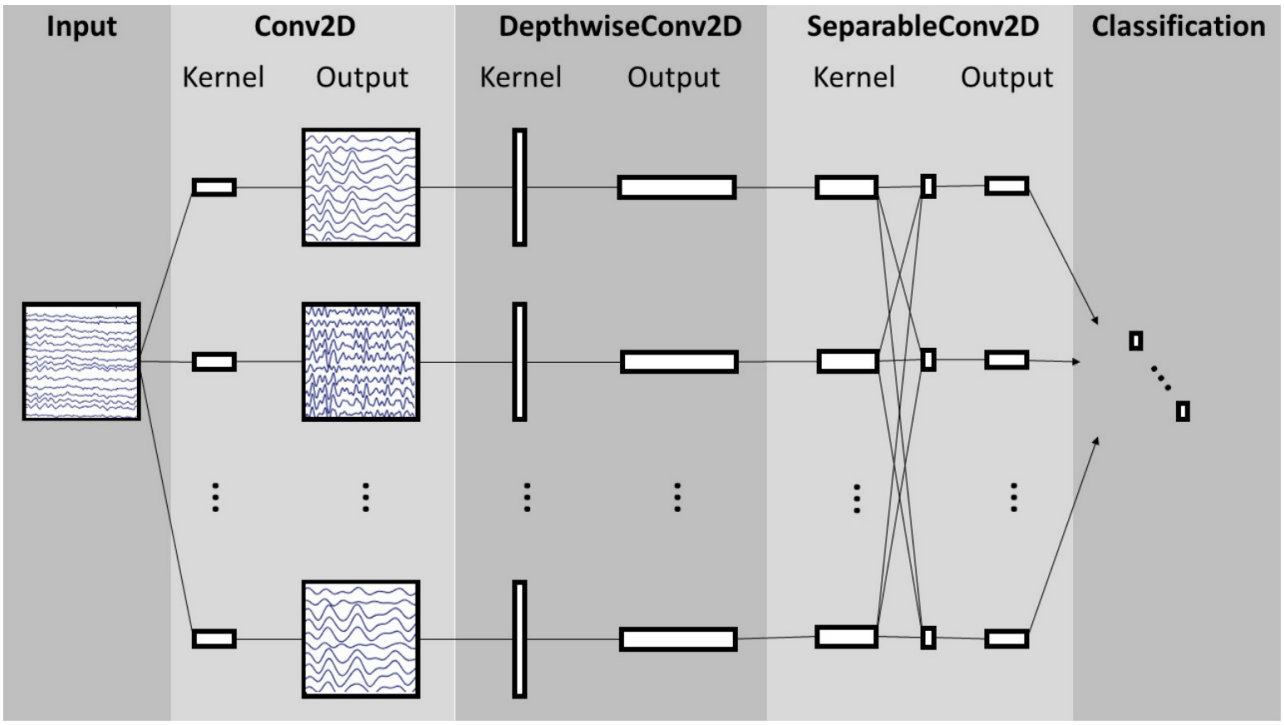
1. Introduction（20%）
在這個實驗中，需要用 pytorch 實作出 EEGNet 和 DeepConvNet 兩種分類模型，應用於 BCI（Brain-Computer Interfaces）competition 資料集。並嘗試三種不同的 activation function（ReLU, Leaky ReLU, ELU）。目標是將腦波訊號分類成兩種類別。
最後需要將實驗結果視覺化表現，畫出 epoch 對精確度的曲線，並進行兩種 model 架構和三種 activation function 的比較討論。

資料集中有兩個 channel，各有 750 個資料點，標記為左,右手兩種類別。



DeepConvNet 就是個傳統的 CNN 架構。

EEGNet 是一種輕量化的捲積網路架構。精簡了許多參數的使用，大幅提昇訓練速度。並且能用少量的訓練資料就得到不錯的結果。EEGNet 的架構大致可以分成四層。先用 Conv2D 學習 frequency filters。接著將輸出個別接上 DepthwiseConv2D，用於學習 frequency-specific spatial filters。然後是 SeparableConv2D，一個深度神經網路，用於統整先前的結果。



2. Experiment set up (30%)

A. The detail of your model

(a) EEGNet

Block	Layer	# filters	size	# params	Output	Activation	Options
1	Input				(C, T)		
	Reshape				(1, C, T)		
	Conv2D	F_1	(1, 64)	$64 * F_1$	(F_1, C, T)	Linear	mode = same
	BatchNorm			$2 * F_1$	(F_1, C, T)		
	DepthwiseConv2D	D * F_1	(C, 1)	$C * D * F_1$	(D * F_1, 1, T)	Linear	mode = valid, depth = D, max norm = 1
	BatchNorm			$2 * D * F_1$	(D * F_1, 1, T)		
	Activation				(D * F_1, 1, T)	ELU	
	AveragePool2D		(1, 4)		(D * F_1, 1, T // 4)		
	Dropout*				(D * F_1, 1, T // 4)		$p = 0.25$ or $p = 0.5$
2	SeparableConv2D	F_2	(1, 16)	$16 * D * F_1 + F_2 * (D * F_1)$	(F_2, 1, T // 4)	Linear	mode = same
	BatchNorm			$2 * F_2$	(F_2, 1, T // 4)		
	Activation				(F_2, 1, T // 4)	ELU	
	AveragePool2D		(1, 8)		(F_2, 1, T // 32)		
	Dropout*				(F_2, 1, T // 32)		$p = 0.25$ or $p = 0.5$
	Flatten				(F_2 * (T // 32))		
Classifier	Dense	N * (F_2 * T // 32)			N	Softmax	max norm = 0.25

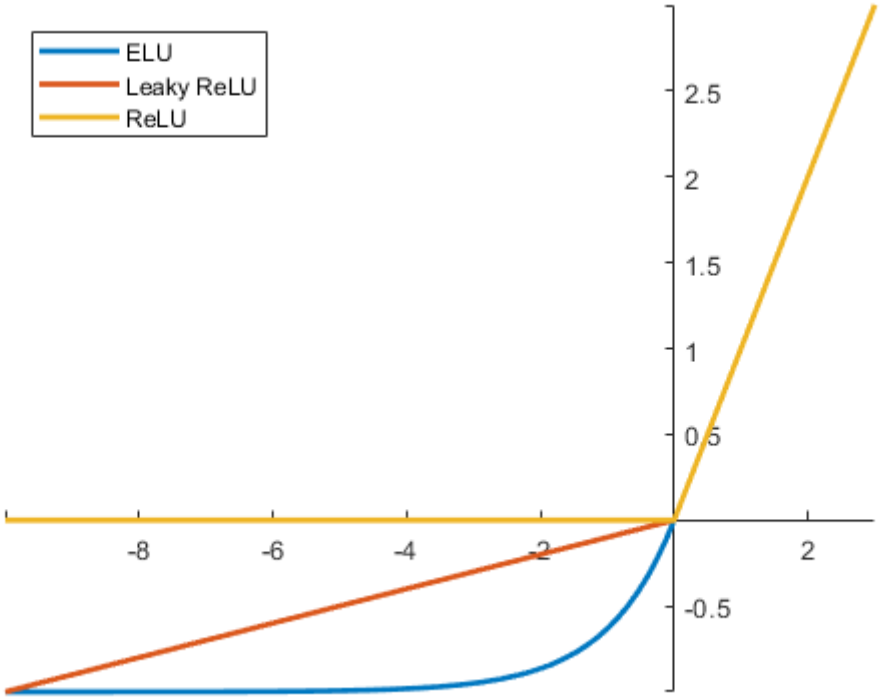
(b) DeepConvNet

Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	50	(1, 5)	25 * 50 * C + 50	Linear	mode = valid, max norm = 2
BatchNorm			2 * 50		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	100	(1, 5)	50 * 100 * C + 100	Linear	mode = valid, max norm = 2
BatchNorm			2 * 100		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	200	(1, 5)	100 * 200 * C + 200	Linear	mode = valid, max norm = 2
BatchNorm			2 * 200		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Flatten					
Dense	N			softmax	max norm = 0.5

Table 5: DeepConvNet architecture, where C = number of channels, T = number of time points and N = number of classes, respectively.

(c) 超參數設定
epoch : 500
learning rate : 0.001275
batchsize : 239
optimizer : Adam
loss function : torch.nn.CrossEntropyLoss()

B. Explain the activation function (ReLU, Leaky ReLU, ELU)



(a) ReLU 的全稱是 Rectified Linear Unit
f(x) = max(0, x)
優點
1. 沒有飽和區，不存在梯度消失問題。
2. 沒有複雜的指數運算，計算簡單、效率提高。
3. 實際收斂速度較快，大約是 Sigmoid/tanh 的 6 倍。
4. 比 Sigmoid 更符合生物學神經激活機制。
缺點
當 x<0 時，ReLU 輸出總為零。該神經元輸出為零，則反向傳播時，權重、參數的梯度橫為零，造成權重、參數永遠不會更新，即造成神經元失效，形成了「死神經元」。

(b) Leaky ReLU 是 ReLU 的改良，將 ReLU 神經元初始化為正偏值，例如 0.01。
f(x) = max(0.01x, x)
優點
1. 沒有飽和區，不存在梯度消失問題。
2. 沒有複雜的指數運算，計算簡單、效率提高。
3. 實際收斂速度較快，大約是 Sigmoid/tanh 的 6 倍。
4. 不會造成神經元失效，形成了「死神經元」。

(c) ELU（Exponential Linear Units）是 ReLU 的另一種改良
f(x) = max(0.01(exp(x)-1), x)
優點
1. 沒有飽和區，不存在梯度消失問題。
2. 沒有複雜的指數運算，計算簡單、效率提高。
3. 實際收斂速度較快，大約是 Sigmoid/tanh 的 6 倍。
4. 不會造成神經元失效，形成了「死神經元」。
5. 輸出均值為零
6. 負飽和區的存在使得 ELU 比 Leaky ReLU 更加健壯，抗噪聲能力更強。

缺點
ELU 包含了指數運算，存在運算量較大的問題。

3. Experimental results (30%)

A. The highest testing accuracy (train/test) (單位：%)

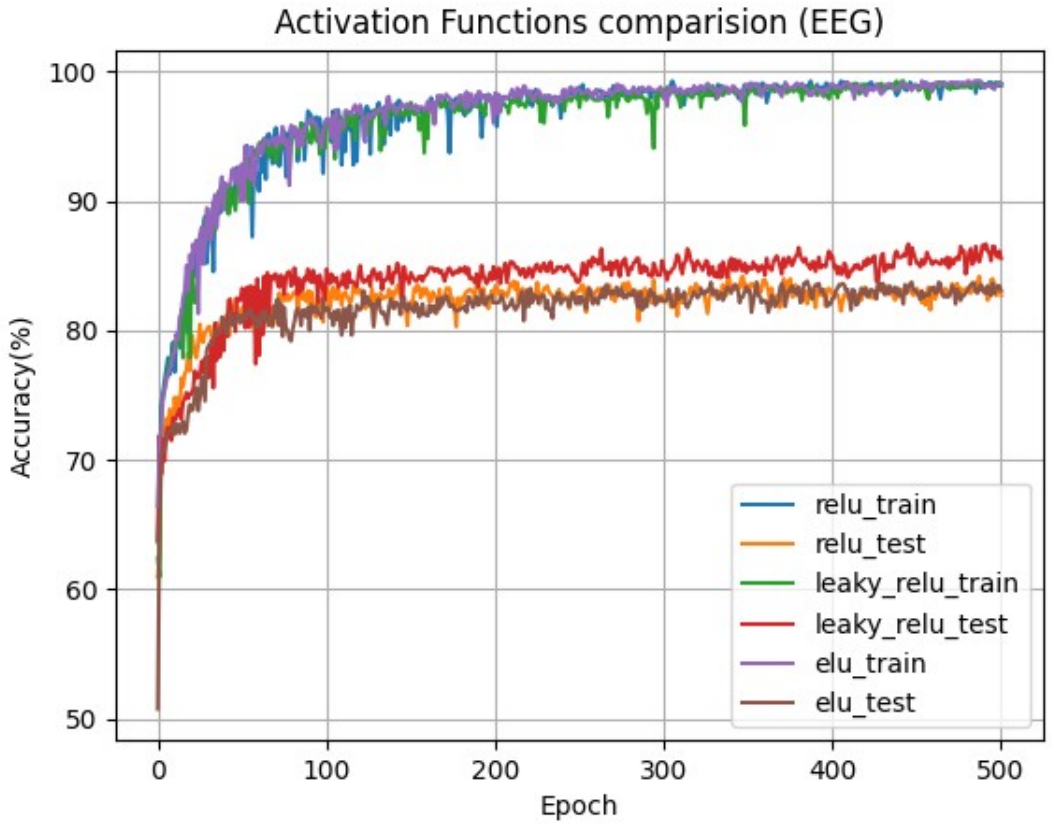
model \ activation	ReLU	LeakyReLU	ELU
EEGNet	98.89/82.69	99.07/85.56	99.07/83.06
DeepConvNet	97.59/79.63	97.59/78.33	97.78/79.71

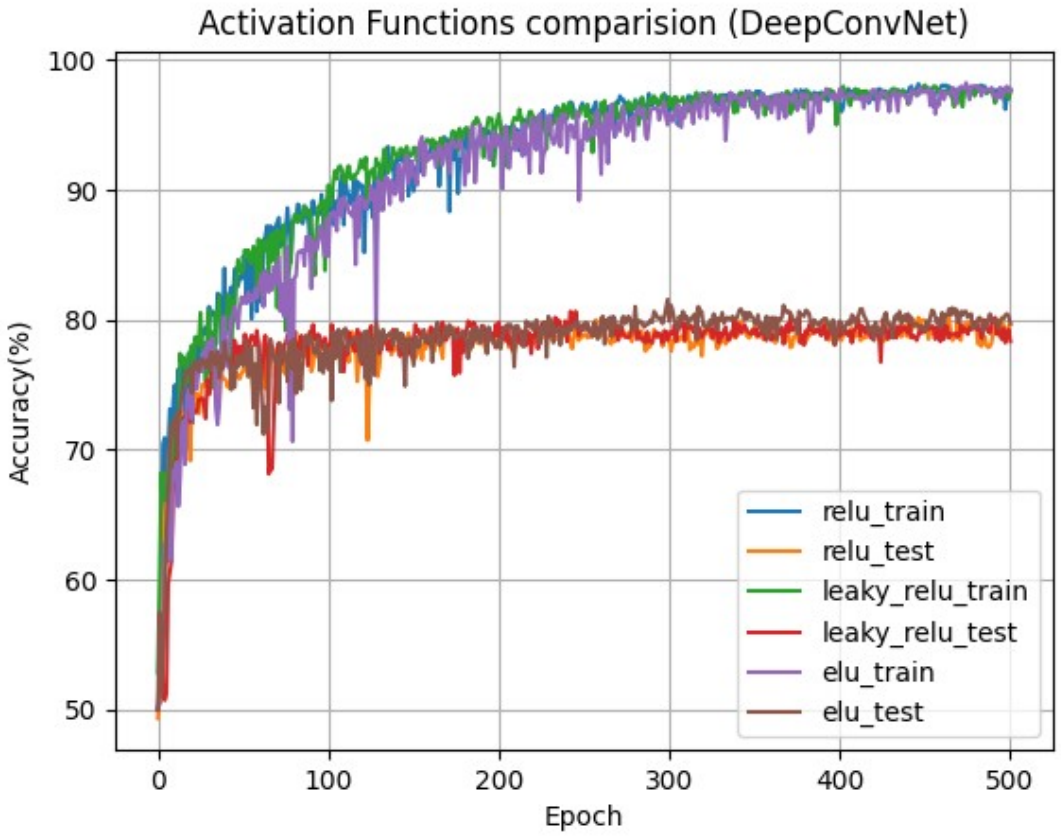
Screenshot with two models

```
yellow@yellow-arg7:~/deep-learning-and-practice/hw3$ python3 hw3.py
(1080, 1, 2, 750) (1080,) (1080, 1, 2, 750) (1080,)
EEG_ReLU_train
epoch=  0  loss=  0.0005947172641754151  correct=  0.6361111111111111
epoch=  250  loss=  5.850958189478627e-05  correct=  0.9879629629629629
epoch=  500  loss=  6.216151156911143e-05  correct=  0.9888888888888889
EEG_ReLU_test
epoch=  0  loss=  0.0006112765382837366  correct=  0.6092592592592593
epoch=  250  loss=  0.0005749557857160215  correct=  0.8305555555555556
epoch=  500  loss=  0.0006904258220284073  correct=  0.8268518518518518
EEG_LeakyReLU_train
epoch=  0  loss=  0.0006148645723307575  correct=  0.6240740740740741
epoch=  250  loss=  0.0001269111754717650 3  correct=  0.9722222222222222
epoch=  500  loss=  4.976184141856653e-05  correct=  0.9907407407407407
EEG_LeakyReLU_test
epoch=  0  loss=  0.0005989440061427929  correct=  0.637962962962963
epoch=  250  loss=  0.0005528010703899243  correct=  0.8379629629629629
epoch=  500  loss=  0.0005143377516004774  correct=  0.8555555555555555
EEG_ELU_train
epoch=  0  loss=  0.0006001835619961774  correct=  0.6638888888888889
epoch=  250  loss=  7.364989982710944e-05  correct=  0.9851851851851852
epoch=  500  loss=  4.445694869867078e-05  correct=  0.9907407407407407
EEG_ELU_test
epoch=  0  loss=  0.0006463523264284487  correct=  0.5074074074074074
epoch=  250  loss=  0.0006644140239115114  correct=  0.8203703703703704
epoch=  500  loss=  0.0006147056266113563  correct=  0.8305555555555556
```

```
DeepConvNet_ReLU_train
epoch=  0  loss=  0.0006839388498553524  correct=  0.5
epoch=  250  loss=  0.0001498338662915759 5  correct=  0.9638888888888889
epoch=  500  loss=  0.0001112114155182132  correct=  0.975925925925926
DeepConvNet_ReLU_test
epoch=  0  loss=  0.000676190577171467  correct=  0.4935185185185185
epoch=  250  loss=  0.0005832577193224872  correct=  0.7888888888888889
epoch=  500  loss=  0.0006872838294064557  correct=  0.7962962962962963
DeepConvNet_LeakyReLU_train
epoch=  0  loss=  0.0006333546506034003  correct=  0.5287037037037037
epoch=  250  loss=  0.0001516798856081786  correct=  0.9629629629629629
epoch=  500  loss=  0.000107426175640689  correct=  0.975925925925926
DeepConvNet_LeakyReLU_test
epoch=  0  loss=  0.0009260321104968036  correct=  0.5
epoch=  250  loss=  0.0005810005245385347  correct=  0.7925925925925926
epoch=  500  loss=  0.0007065689122235333  correct=  0.7833333333333333
DeepConvNet_ELU_train
epoch=  0  loss=  0.0008268872344935382  correct=  0.5009259259259259
epoch=  250  loss=  0.0001784800517338293 6  correct=  0.95
epoch=  500  loss=  8.164019910273728e-05  correct=  0.9777777777777777
DeepConvNet_ELU_test
epoch=  0  loss=  0.0007099973382773222  correct=  0.5
epoch=  250  loss=  0.0005817222926351759  correct=  0.7944444444444444
epoch=  500  loss=  0.0006630972027778625  correct=  0.7972222222222223
```

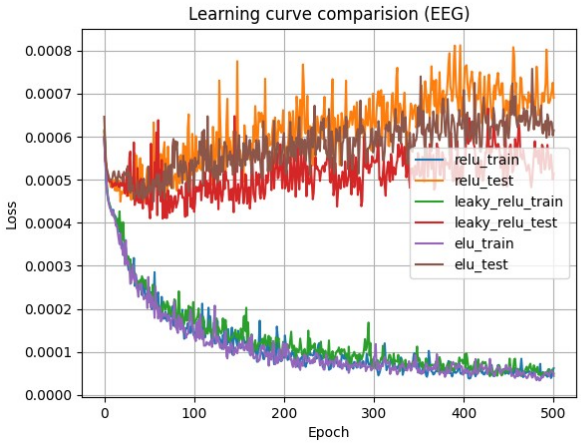
B. Comparison figures

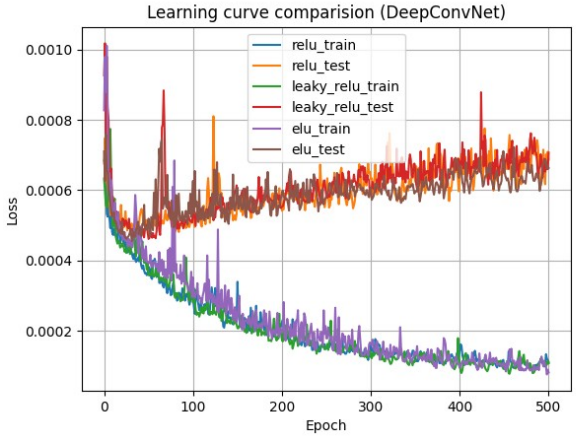




4. Discussion（20%）

1. 發現大概訓練 100 個 epoch 後，在 train data 上就已經可以達到約 97%正確率，但是在 test data 上繼續訓練也沒有太多進步。覺得某種程度上它可能已經 overfit 了，從 Loss 的學習曲線中也可以看出來。





2. 因為訓練到後期，精確率無法進步，嘗試用 StepLR（torch.optim.lr_scheduler.StepLR）調整 learning rate，每經過 100 的 epoch 將 learning 乘上 0.985。效果不太顯著。

3. 雖然資料量不大，但如果不使用 mini batch，正確率大概只能 70%左右。使用了 mini batch 後大幅上升到 80%以上。