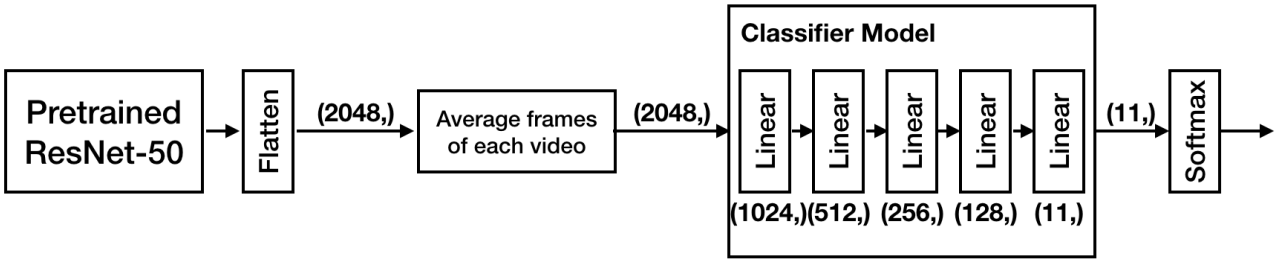
名字： 劉兆鵬 系所：電機所碩一 學號：R07921052

Problem 1: Data preprocessing

1. Describe the strategies of extracting CNN-based video features, training the model and other implementation details(which pretrained model) and plot the learning curve.

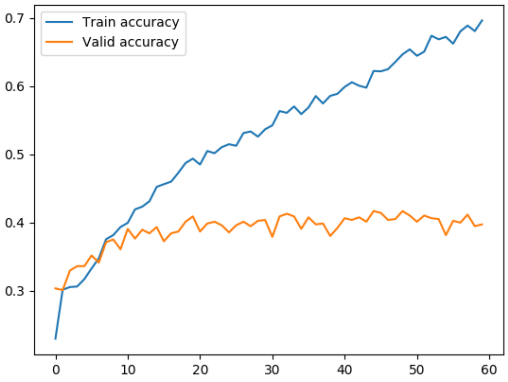
此題中我為了避免在訓練時將 video 的每一個 frame 都丟入 pretrained model 取 feature 時會導致 gpu out of memory 的問題，故在此我先使用一個 encoder 先將所有影片的 frame 都先經由此 pretrained model 將 feature 取出後儲存起來，而在此我所使用的 pretrained model 為 ResNet50，使之能夠在訓練 classifier 時能夠直接使用，而不會導致 memory 不足夠的問題。而在萃取 frame 的 feature 時，為了能夠將我的使用的 data 符合 pretrained model 的 distribution，故我先將每一個 frame 根據其 pretrained 時所使用的 scale 進行 normalize，再進行 feature extraction，使之能夠正確的取出對應的特徵。且為了增加 training data 的多樣性，我將每一個影片的 frames 都使用 flip 後加入 data 中一同 encode 其 feature，使 data 數量增加兩倍。

而將 feature 取出後，在訓練的過程中由於每一個影片的長度不同，故在讀取影片時，我會紀錄每一個影片各自的長度，以便在進入 classifier model 時能夠取得對應的 frame。而為了能夠讓每一個影片出來的 feature 維度相同，故我對每一個影片的 frames 取 mean，使一個影片會得到一個 global 的 feature，並將之送入 classifier model 進行訓練。而我的 classifier model 架構如下所示。

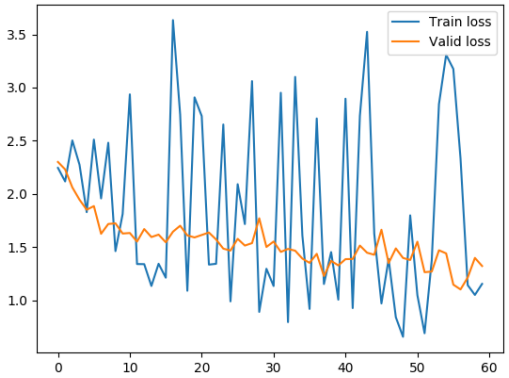


Model architecture

透過此 classifier model 則能將影片的 feature 輸出至 class number 數量進行訓練。而在訓練過程中的 learning curve 如下所示，可以得知 validation accuracy 到一定程度後就不會繼續的提升，而 loss 也是不再繼續下降，為了避免 overfitting，故我採用 early stop 選取適當的結果作為最後的 model。



Accuracy



Loss

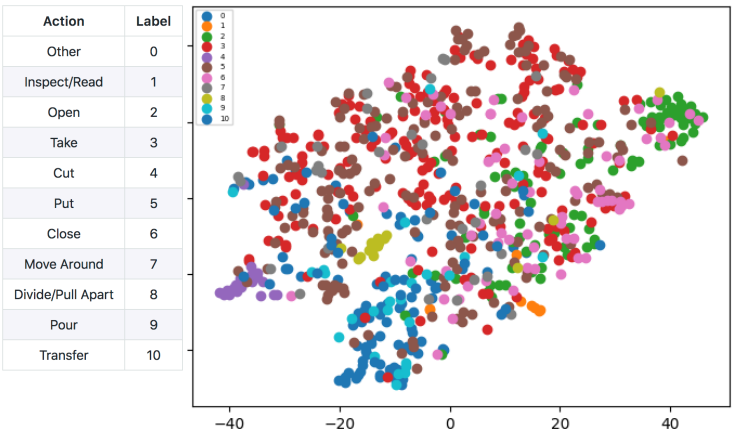
2. Report the video recognition performance (valid) using CNN-based video features.

在將 classifier model 也訓練完成後，最後我將 validation data 的影片分別讀入經由 pretrained ResNet 50 以及訓練後的 classifier model 進行 predict。其準確率如下所示。

	CNN-based
Accuracy	0.4486

3. Visualize CNN-based video features to 2D space (with tSNE) in the report.

接著我將訓練後的 model 輸入 validation video 後，在 hidden layer 的最後一層中將此 feature 取出，經由 tSNE 把 video 的 feature 投影至二維空間中，以便繪出結果分佈，並將每一個影片的節點根據其 label 使用不同顏色進行區隔，使圖能夠顯示出其測試分佈。其結果如下圖所示。

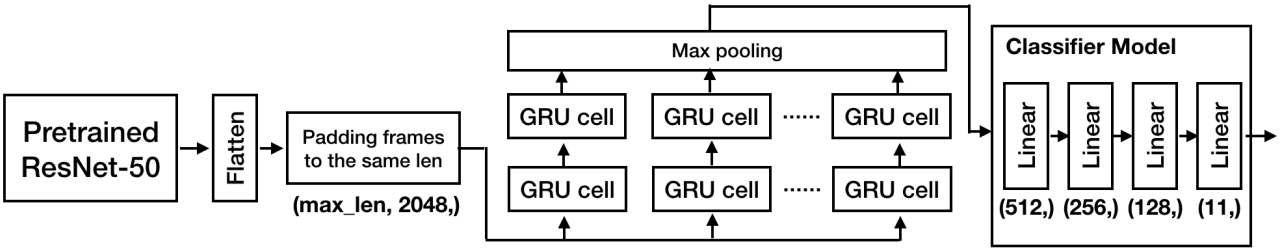


Action	Label
Other	0
Inspect/Read	1
Open	2
Take	3
Cut	4
Put	5
Close	6
Move Around	7
Divide/Pull Apart	8
Pour	9
Transfer	10

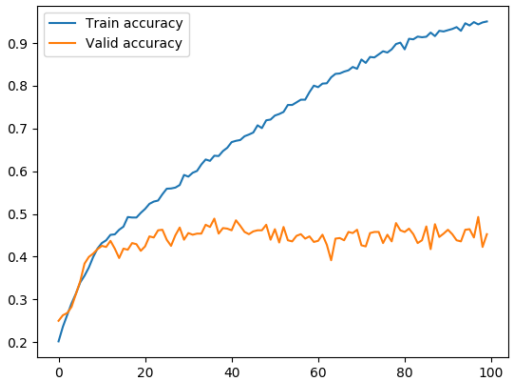
Problem 2: Trimmed action recognition

1. Describe the RNN models and implementation details for action recognition and plot the learning curve of the model.

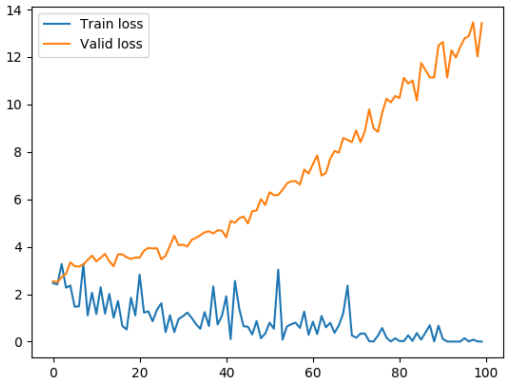
在此題我使用與前一題想同的 feature extraction 方法，故我將影片的 feature 都萃取出後就可以使用 RNN-based model 進行 classify。接著由於在使用 RNN 進行 batch 的訓練時，需要讓 batch 中每一個 sequence length 一樣長，故在此我取出此 batch 最長的 video 長度，並將此 batch 中的每一個 video 都 zero padding 成相同的長度後，再傳入 RNN model 中進行計算。而在此我使用的 RNN model 為 GRU 且使用兩層 bidirectional 進行 feature 運算。在 RNN 出來的結果也是為 sequence length 長度的 feature，故我使用 max 取出每一個 element 最具意義的 feature 出來使之合成 global feature 後再經由 Linear 進行分類，此 RNN classifier model 架構如下圖所示。



透過此 RNN-based classifier model 則能將影片的 feature 輸出至 class number 數量進行訓練。而在訓練過程中的 learning curve 如下所示，可以得知 validation accuracy 到一定程度後就不會繼續的提升，而 loss 也是不再繼續下降，為了避免 overfitting，故我採用 early stop 選取適當的結果作為最後的 model。



Accuracy



Loss

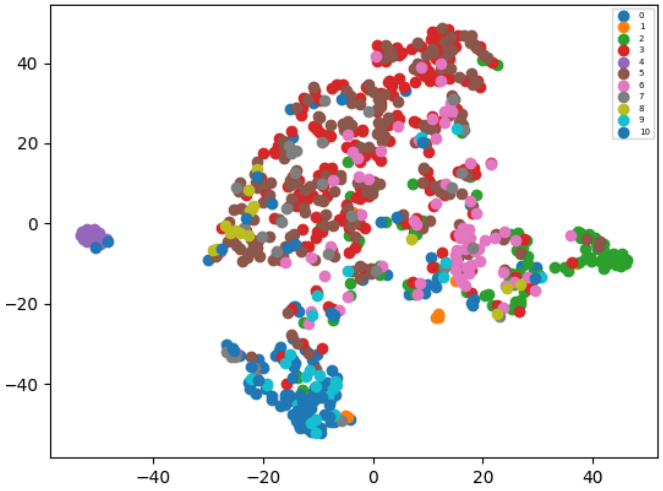
2. Report the video recognition performance (valid) using RNN-based video features

在將 classifier model 也訓練完成後，最後我將 validation data 的影片分別讀入經由 pretrained ResNet 50 以及訓練後的 classifier model 進行 predict。其準確率如下所示。

	RNN-based
Accuracy	0.4980

3. Visualize RNN-based video features to 2D space (with tSNE) in the report. Do you see any improvement for action recognition compared to CNN-based video features? Explain.

接著我將訓練後的 model 輸入 validation video 後，在 Linear hidden layer 的最後一層中將此 feature 取出，經由 tSNE 把 video 的 feature 投影至二維空間中，以便繪出結果分佈，並將每一個影片的節點根據其 label 使用不同顏色進行區隔，使圖能夠顯示出其分佈。其結果如下圖所示。



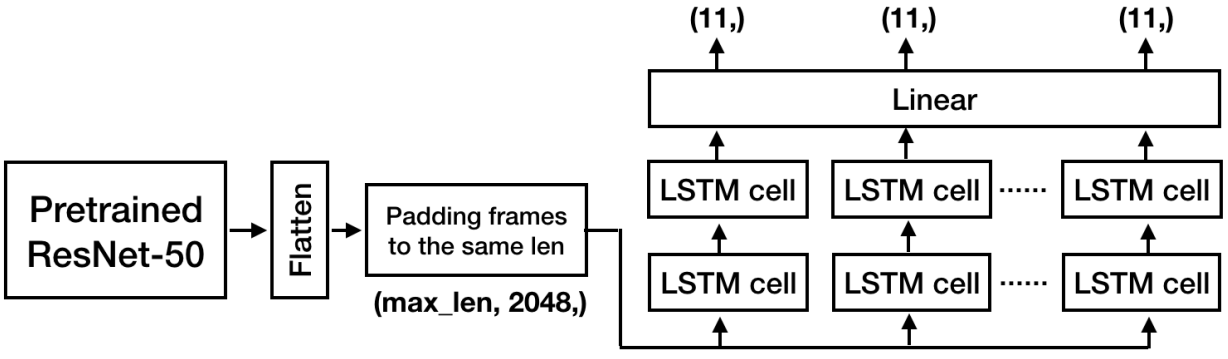
Action	Label
Other	0
Inspect/Read	1
Open	2
Take	3
Cut	4
Put	5
Close	6
Move Around	7
Divide/Pull Apart	8
Pour	9
Transfer	10

從上題兩者的 tSNE 結果可以發現兩個不同 model 所輸出的分佈集中度將當不同，在 CNN-based feature 中，每個類別的節點並沒有辦法完善的集中在一起，且所有的資料分佈非常地分散，這也使得在最後 classifer layer 無法準確的將 feature 準確的分類出來。然而使用 RNN-based feature 卻能夠將各個不同類別資料分散的較開，使得 classifier layer 能夠較好的將各個類別分類出來。雖然在 RNN-based feature 尚有一些類別較難被區分出來，但效果卻已經比 CNN-based feature 要來得好。我認為是因為在使用 CNN-based feature 時，是將每個影片 frame 的 feature 直接取平均，並沒有去理解相同影片中每個 frame 彼此之間的關係。導致只使用 CNN-based feature 無法讓 model 去了解到 frames 彼此的先後關係來取得更佳的 video feature。但在使用 RNN-based feature 上，因為使用 RNN 將每個影片的 frames 萃取出彼此之間先後順序的 feature，故能夠透過前後關係來更容易的推測出，此 video 的類別為何，故在最後的資料分佈上取得 RNN-based feature 較 CNN-based feature 佳的結果。

Problem 3: Temporal action segmentation

1. Describe any extension of the RNN models, training tricks, and post-processing technique you used for temporal action segmentation.

在此題中，我也與前兩題使用相同的方式先將影片中每一個 frame 的 feature 先萃取出來，然而不同是在此部分助教已經幫我們將影片轉成圖片的格式了，故我根據每一個 category 將其圖片的 feature 萃取出後儲存起來。而由於在訓練時，無法將影片內容中所有的 frame 一次一起放入 classifier model 中進行訓練，且在此題需要將每個 frame 的 action 都分類出來，故在此我採用切割 video 的方式，一次只取一部影片 350 個 frame 作為一個 data，故將所有影片都以每 350 個 frame 進行切割，並保留 50 個 frame 的 overlap，使得一部影片會被切割為多個 training data。而我在訓練的 classifier model 與前一題的 RNN-based feature model 相似但採用 bidirectional LSTM unit 作為我的 RNN layer，其架構如下所示。而由於 training videos 中每一個 action 的數量較不平均使得 training data imbalance 的問題，在此我採用 weighted loss 的方式進行訓練，給予數量較少的類別較高的權重，使得訓練的結果能夠提升。而最後再測試時，則是將影片一部一部讀進後使用 pretrained ResNet50 以及訓練後的 classifier 對每個 frame 進行預測，而由於每個 frame 都需要被預測出一個類別，而在連續的 frame 中，不可能在兩個前後相同類別的 frame 之間包含一下不同類別的動作，故我在此判斷若前後兩者 frame 為相同類別時，則中間的 frame 改為與前後將同的動作，用此方式最為我的 post-processing 的方法，也能夠使準確度提升。

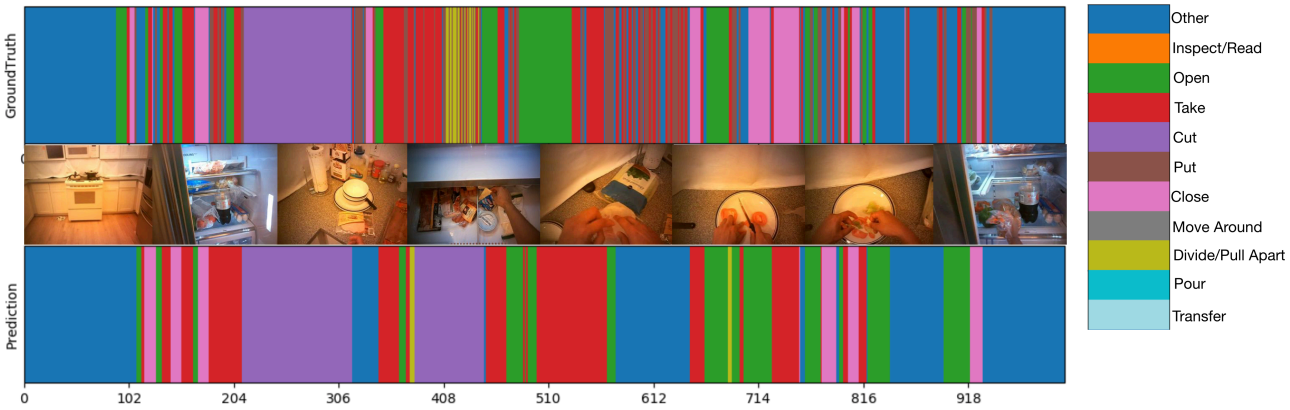


2. Report validation accuracy in the report.

	RNN-based frame by frame
Accuracy	0.5940

3. Choose one video from the 7 validation videos to visualize the best prediction result in comparison with the ground-truth scores in the report.

在此題中我將每個 frame predict 出各自的類別為何，在此我選擇 *Op01-R02-TurkeySandwich* 這部影片作為我的結果展示，如下圖所示。圖中下欄為其 predict 出來的結果，上欄則為 ground truth 的 label，中間則為根據時間軸所挑選出來得 video frames。從圖中可以發現在藍色類別的部分可以被預測的較為準確，因為那個類別在訓練資料中佔了絕大部分的數量，使得 Model 比較能夠判斷出那個類別，但也因此導致類別數量較少的類別無法被準確的預測出來結果，雖然已經使用了 weighted loss，但依然效果有限。而因為有時做 post-processing 的部分所以可以使 predict 的結果大部分都是連續的類別相連。最後從下圖可以觀察出來，若是能夠增加或是平均每個類別的訓練資料，是有機會能夠讓訓練結果達到更好的。



Problem 4: Bonus

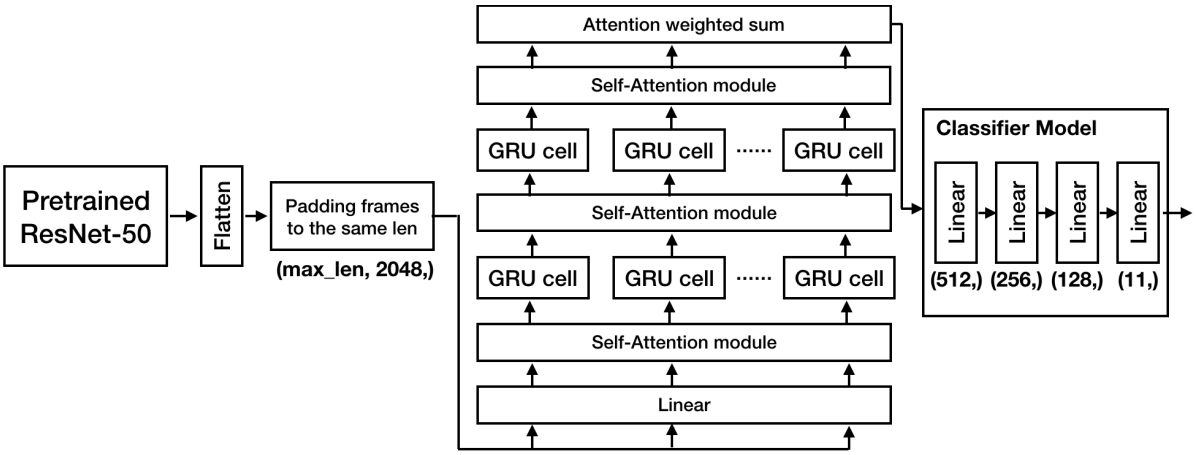
1. Implement and integrate attention mechanisms to improve both trimmed action recognition and temporal action segmentation.

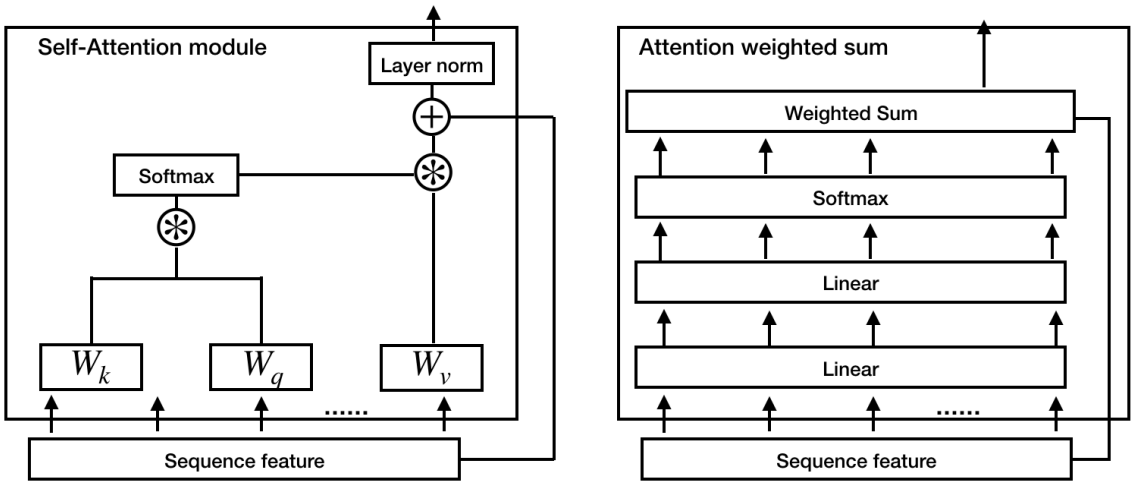
在此題中，我實作了 attention 的 model 並將之方法加入至 problem 2 以及 problem 3 中，使得其 performance accuracy 能夠提高一些。其各自的方法如下列所述。

Problem 2:

此題中的 feature extraction 以及訓練方式都與第二題所述相同，唯一改變的地方為將原先的 model 加入了 Attention 的方法。方法為在將 feature 萃取出來後，將各個 video 的 frame 根據其 batch 最長的長度做 zero padding，並使用 linear projection 將 feature 投影至較低維度的特徵。接著我使用 Self-Attention 的方法，其為讓 sequence 中的每個 feature 對自己 sequence 中的每個 feature 取出 Attention weight 的方法，我先將 sequence 中的每個 feature 使用三個不同的 linear layer 取出 query, key 以及 value 的特徵資訊，接著我使用 query 去對 key 做 dot 的計算，並對其做 softmax，則表示 sequence 中的每一個 feature 對其他 feature 所需要注意的權中為何，接著將其權重與 value 做 weighted sum，則能得到每一個 time step 的 feature 與其他 time step feature 的加權運

算，使得此 time step 的輸出並不是此從自己的 frame 得來，而是會根據其他 time step 的資訊來 encode 出新的 feature，接著我將原先 attention 之前的 feature 與 attention 後的 feature 相加並做 layer normalization 將 feature 做 skip connection。再將此 sequence feature 通過兩層 bidirectional GRU，而在每一層的輸出都會接上與上述相同的 self-attention，skip connection 以及 layer normalization 的方法。最後因為所有的 frame 只為表達一個類別的動作，故需要將 sequence 中所有 time step 的 feature 進行結合，而在此我採用一般的 attention 方式，將所有 time step 的 feature 接上 linear layer 並使用 softmax 算出其各自的權重值後，對所有 time step 的 feature 做 weighted sum 的運算，最後再將此得出的 feature 送入與第二題相同的 Linear layer 進行 action label 的訓練。其 model 架構以及 performance accuracy 如下所示。

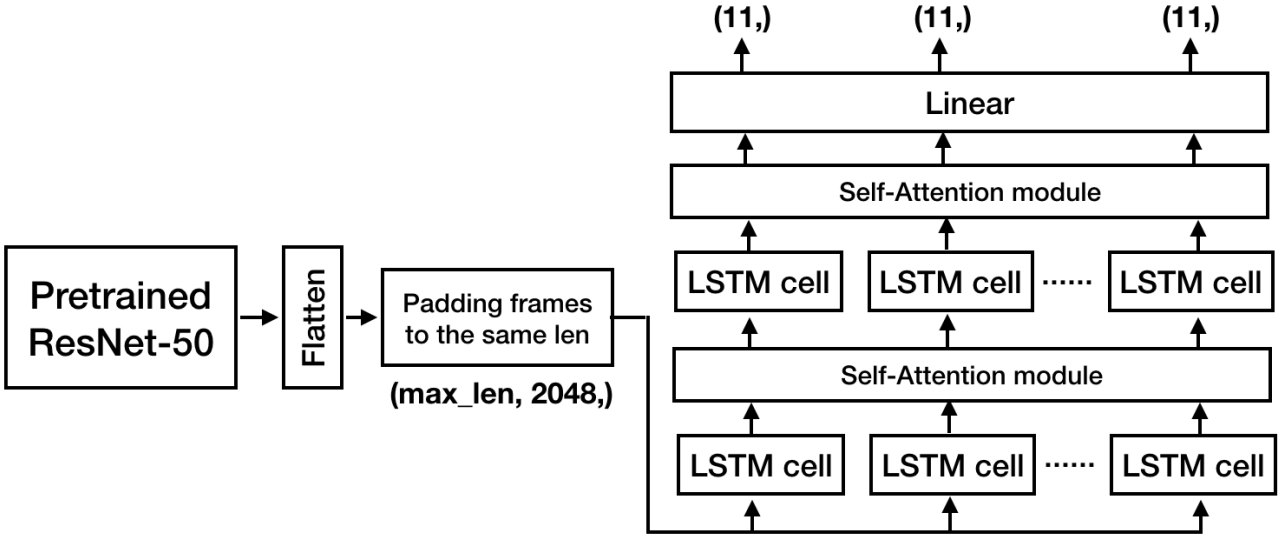


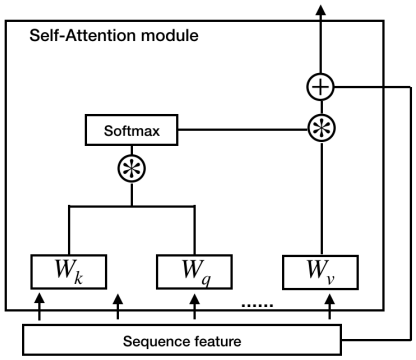


	RNN-based	RNN-based with Attention
Accuracy	0.4980	0.5201

Problem 3:

此題中的 feature extraction 以及訓練方式都與第三題所述相同，唯一改變的地方為將原先的 model 加入了 Attention 的方法。方法為在將 feature 萃取出來後，也採用將每個 frame 擷取最大長度為 350 個 frame 為一個 data 進行訓練，在 model 運算為，首先我將 videos 的 frames 先經由 LSTM 出來後進入 Self-Attention module 計算每個 time step feature 彼此之間的關係，在此也加入 skip connection 使原資訊能夠傳入至 self-attention 所 encode 出的結果。其 Self-Attention module 內的運算也為使用三個不同的 linear layer 取出 query, key 以及 value 的特徵資訊，接著使用 query 去對 key 做 dot 的計算，並對其做 softmax，則表示 sequence 中的每一個 feature 對其他 feature 所需要注意的權中為何，接著將其權重與 value 做 weighted sum，則能得到每一個 time step 的 feature 與其他 time step feature 的加權運算。此作法在第一層以及第二層 LSTM 的輸出皆會使用此 Self-Attention module 來 encode sequence time step 的 feature。最後因為每一個 frame 都要能夠輸出其對應的動作類別，故在此不需要將每個 time step 的 feature 進行結合，只需將每一個 time step feature 接上與 problem 3 相同 linear layer 來進行 classify。其 model 架構以及 performance accuracy 如下所示。





	RNN-based frame by frame	RNN-based frame by frame w/ Attention
Accuracy	0.5940	0.6058