Final Proposal

隊名及隊員

- 隊名：老司機帶我飛
- 隊員：

學號	姓名
B04902131	黃郁凱
B04902019	王士弘
B05902109	柯上優

選擇題目：

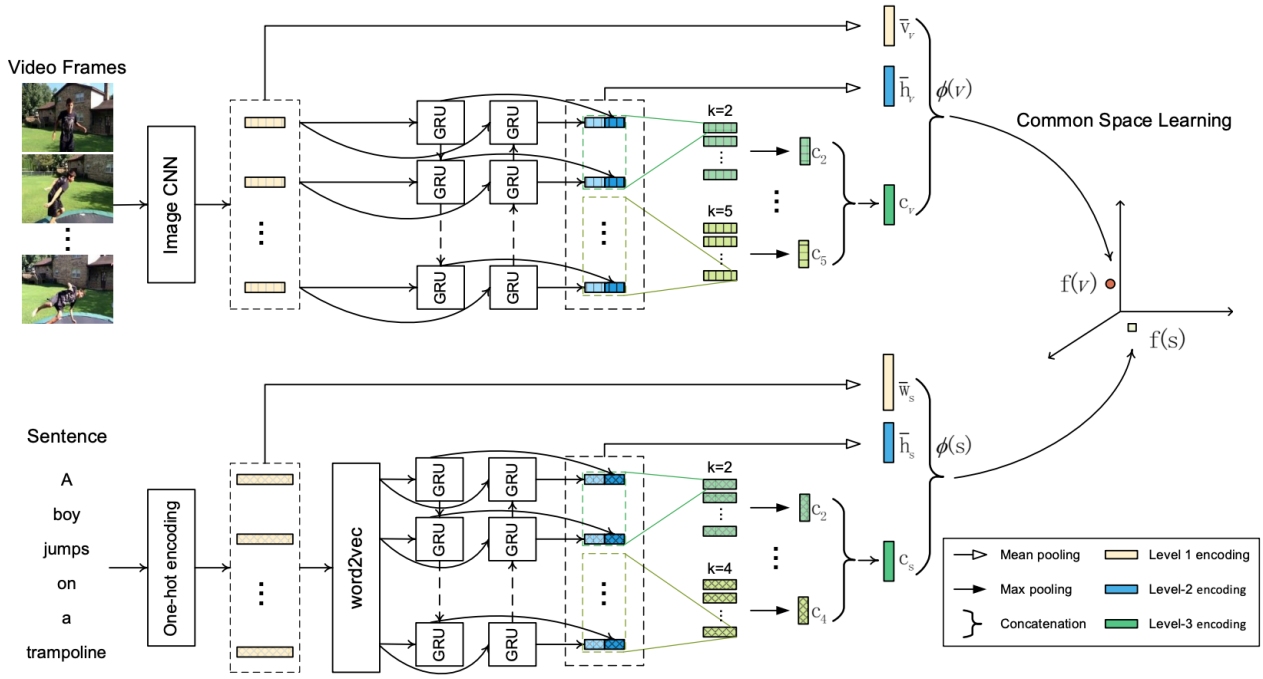
- Video Caption

Problem Study

論文一

方法與模型架構

此篇論文的目標是實作 “Zero-Example Video Retrieval”，input 為一段文字敘述，output 與敘述相關的影片。輸入沒有任何 visual example 作為參考，因此稱為 Zero-Example 。模型架構如圖：



video 和 query 會先分別被 encode 成向量 $\phi(v)$ 和 $\phi(s)$。video 的 encode 方法如下：

- 從 video 中抽取 n 個 frames，先利用 Image CNN 抽取 video frames 中的 deep features，經過 mean pooling 後作為 $\bar{v}$。
- 將 $\bar{v}$ 作為 biGRU 的輸入 (一層為 forward GRU，另一層為 backward GRU)，將其各 step 的 hidden state 串接成一 feature map，經過 mean pooling 後作為 $\bar{h}$。
- 將 feature map 作為五個 1-d CNN 的輸入，經過 ReLU 後再經過 max pooling，將五個結果串接成 c。
- 將上述的 $\bar{v}, \bar{h}, c$ 串接成為 $\phi(v)$，即是 video encode 過後的 vector。

query 的 encode 方法如下：

- 對 query 做 one-hot encoding，其結果作為 $\bar{w}$。
- 利用 word2vec 產生 biGRU 的輸入，利用上述相同的方法得到 $\bar{h}$ 和 c。將 $\bar{w}, \bar{h}, c$ 串接成為 $\phi(w)$，即是 query encode 過後的 vector。

計算 encode 後vector的相似程度，使用 common space learning:

- 將 $\phi(v)$ 和 $\phi(w)$ 經過一層linear轉換後映射到同一個 space 中，並做batch normalization
- 用 cosine similarity 計算相似度，並計算 triplet ranking loss 以決定與 query 最相關的 video
- 詳細數學如下：
  - $f(v) = BN(\phi(v) \cdot W_v + b_v)$
  - $f(w) = BN(\phi(w) \cdot W_w + b_w)$
  - Distance between $f(v)$ and $f(w)$ $S(f(v), f(w)) = \frac{f(v) \cdot f(w)}{\|f(v)\| \|f(w)\|}$
  - Triplet loss:
    $max(0, \alpha + S(f(v_p), f(w_n)) - S(f(v_p), f(w_p))) + max(0, \alpha + S(f(v_n), f(w_p)) - S(f(v_p), f(w$
    - 其中α是margin，n代表negative example，p代表positive example

我們可用的概念

- 此模型架構可以套用在我們的題目上，作法是將 input video feature 和四個 query 經過此模型 encode 後，投射到common space上，並計算與影片最相關的caption當作輸出。

Reference

- Jianfeng Dong, Xirong Li, Chaoxi Xu, Shouling Ji, Xun Wang “Dual Dense Encoding for Zero-Example Video Retrieval”

論文二

方法與模型架構

- 目標是將句子的Word2Vec訓練成Word2VisualVec，讓句子和圖片映射在同一個空間上。

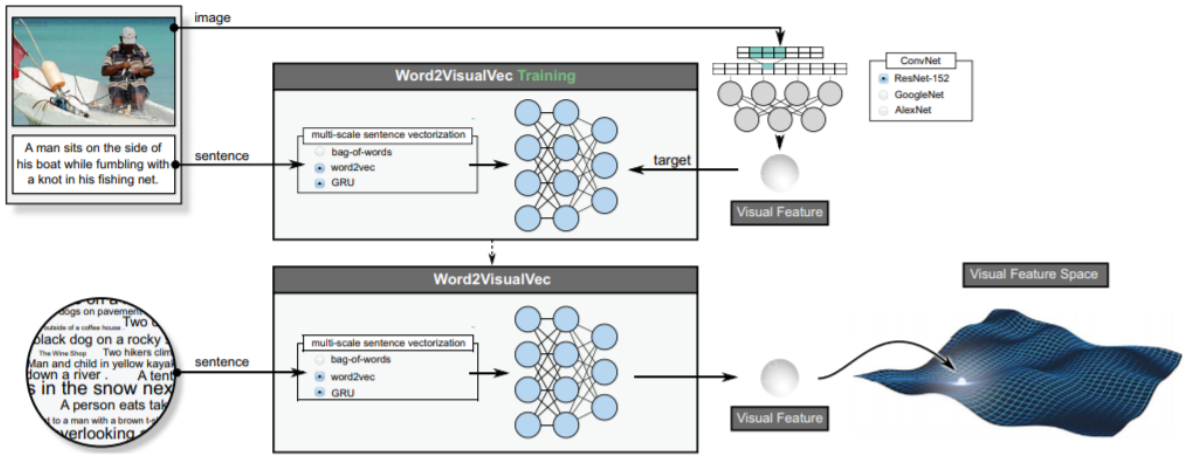


Fig. 2. **Word2VisualVec network architecture**. The model first vectorizes an input sentence into a fixed-length vector by relying on bag-of-words, word2vec and a GRU. The vector then goes through a multi-layer perceptron to produce the visual feature vector of choice, from a pre-trained ConvNet such as GoogleNet or ResNet. The network parameters are learned from image-sentence pairs in an end-to-end fashion, with the goal of reconstructing from the input sentence the visual feature vector of the image it is describing. We rely on the visual feature space for image and video caption retrieval.

- 對於句子q，採用三種方式處理
  - Bag-of-Words: $s_{bow}(q) = ((c_{w_1}, q), (c_{w_2}, q), \ldots, (c_{w_m}, q))$
  - Word2Vec: $s_{word2vec}(q) = \frac{1}{|q|} \sum_{w \in q} v(w)$
  - RNN: use GRU base, determine as $h_{|q|}$
  - 最後concatenate在一起變Multi-scale: $s(q) = (s_{bow}(q), s_{word2vec}(q), h_{|q|})$
- 使用 multilayer perceptron 訓練 $s(q)$ ，將相近的句子在 VisualVec space 上拉近，以 Mean square error 作為 objective function。
- 使用 pretrain CNN 將圖片的 feature 抽出，使圖片的 VisualVec $\phi(x)$ 和配對文字的 VisualVec $r(q)$ 拉近，以 cosine similarity 作為 objective function。以下是使用過的 pretrain model
  - CaffeNet
  - GoogLeNet
  - GoogLeNet-shuffle
  - RestNet-152
- 對於4 × 4種組合方法，各對 Flickr8k 和 Flickr30k 兩種 dataset 做實驗。

我們可用的概念

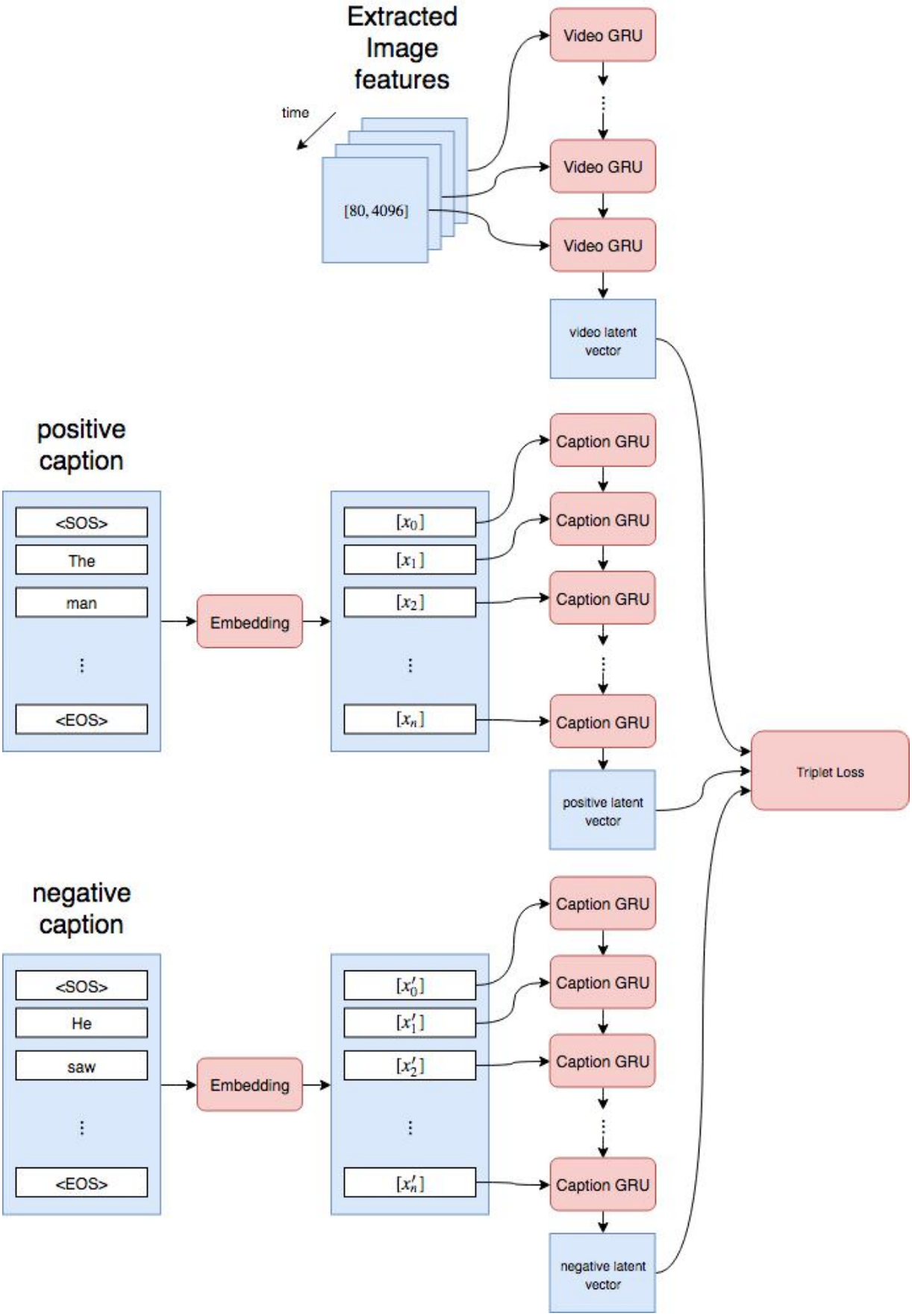
- training data 的圖片，助教已經幫我們抽出 feature ，所以我們可以參考論文前半部的想法，例如 (1.)Word2VisualVec 與 (2.)類似文句VisualVec的訓練。
- Word2VisualVec
  - BoW、Word2Vec、RNN 都是可以抽取caption feature的方法。
- 類似文句VisualVec的訓練:
  - dataSet給了十句都符合圖片的句子，由於我們看不到原始影片，我們便假定這十句都是描述著同一個影片主題，且都是正確的。試著將十句話在 VisualVec space 的位置拉近。

Reference

- Jianfeng Dong, Xirong Li, and Cees G. M. Snoek “Predicting Visual Features from Text for Image and Video Caption Retrieval”

proposed method

現在的模型



- Trainable Parameters
  - embedding
    - 單純將word轉換成vector的參數
    - 大小是$|V| \cdot D$，V是vocabulary的大小，D是embedding dimension

    - Video GRU
        - 負責萃取時間軸相關的影片資訊
        - 一開始 `hidden state` 給None（pytorch會initialize成全0）
        - 最後一個時間點GRU的 `hidden state` 當作濃縮影片的feature vector，稱之 `video latent vector`
    - Caption GRU
        - 萃取一個句子所含有的資訊
        - 一開始 `hidden state` 給None
        - 最後一個時間點的GRU的 `hidden state` 代表句子的feature vector
        - 不管是positive caption（與影片相關的句子）或者negative caption（隨意抽取一個不相關的句子），都會使用同的GRU來encode資訊
- Training Network Architechture
    - Video
        - 已經將每個frame的相片抽好feature，大小是4096
        - 將80個frame依時間順序送入Video GRU中
        - `hidden state` 當作video的feature
    - Caption
        - 送入embedding轉成$L \cdot ED$，L是sequence length，ED是embedding dimension
        - 依句子順序送入Caption GRU當中
        - `hidden state` 當作caption的feature
        - 一次會分別送入兩句，一句是解答，一句是錯誤的句子，Caption GRU分別encode兩個vector
    - Combine
        - 現在有三個大小相同的vector分別是 `video feature`, `positive feature` 和 `negative feature`
        - 使用**Triplet Loss**將 `video feature` 跟 `positive feature` 拉近，將 `video feature` 和 `negative feature` 拉遠
- Inference
    - 將五句分別用caption GRU encode出五個feature vector
    - 找與video feature最相近的當作輸出
- Loss Function
    - Triplet Loss
        - 目的是希望在feature space上可以將相似類型的vector聚集，而不相似的拉遠
        - 令$f(V)$是video的feature vector，$f(C_p), f(C_n)$分別是正確與錯誤句子encode完得vector，α是可調的hyperparameter
        - 目標：
            - $\|f(V) - f(C_p)\|_2 + \alpha \leq \|f(V) - f(C_n)\|_2$
            - 使得vector間的距離大於某個margin（α）

預計改進的方法

- caption抽feature的方法
    - 問題：目前只用RNN的hidden state當feature vector，是否有其他可以取用feature的資訊？
    - 改進方案：如同**論文二**，可以加入Bag of word以及word2vec，或許可以增加performance

- feature vector
  - 問題：直接拿GRU最後一個時間點的hidden state當作featrue vector感覺不夠細膩
  - 改進方案：如果如同**論文一**使用**common space learning**的方法，可以改善此點：所有的feature都要被投射到同一個space中，才繼續計算之後的triplet loss
- triplet loss
  - 問題：目前的做法只有拉近正確與錯誤語句的距離而已，但是如果兩個影片不相關，卻被投射到鄰近的space中該怎麼辦？
  - 改進方案：可以試試看"tetraplet loss"，多加一個不相關的影片encode的vector$f(V_n)$，現在的目標變成有二
    - $\|f(V) - f(C_p)\|_2 + \alpha \leq \|f(V) - f(C_n)\|_2$
    - $\|f(V) - f(C_p)\|_2 + \alpha \leq \|f(V_n) - f(C_p)\|_2$
  - 也就是多加了將錯誤影片與正確句子拉遠的目標
  - 或者更進一步可以直接將兩個影片拉遠:$\|f(V) - f(V_n)\|_2 \geq \alpha$
- triplet loss distance function
  - 問題：目前使用L2 norm來代表兩個vector的距離
  - 改進方案：可以試試其他的function來計算vector的距離，例如cosine similarity, Tanimoto
    - cosine similarity: $cos(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|}$
    - Tanimoto: $T(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|^2 + \|v_2\|^2 - v_1 \cdot v_2}$
  - 或者甚至可以用個係數β, γ來調整之間的比例
    - $Distance(v_1, v_2) = \|v_1 - v_2\|_2 + \beta cos(v_1, v_2) + \gamma T(v_1, v_2)$
- Embedding
  - 使用 `torch.nn.Embedding()` 和語句一起下去訓練，但是可以嘗試pretrain的embedding，效果有機會更好