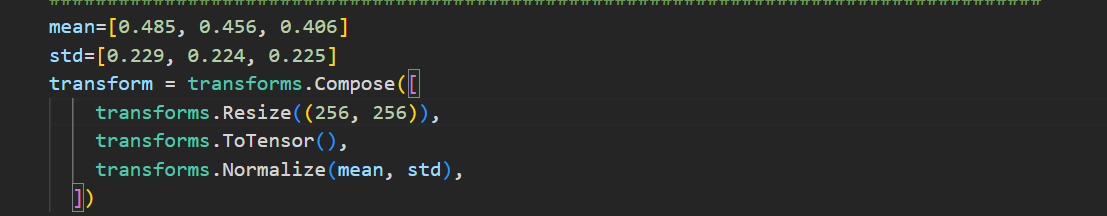
Assignment 6 Part 1: Image Captioning

## Garbage Output\_1

* Data Augmentation



* Encoder Model

\_\_init\_\_()

一張含有 文字 的圖片

自動產生的描述

forward()

一張含有 文字 的圖片

自動產生的描述

* 這裡我使用的pretrained model是**Resnet152**
* Decoder Model

\_\_init\_\_()

一張含有 文字 的圖片

自動產生的描述

forward()一張含有 文字 的圖片

自動產生的描述

* Training Settings

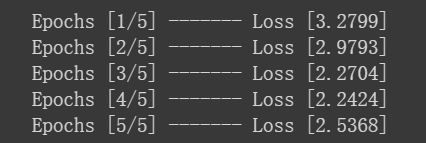
一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

* Training Result

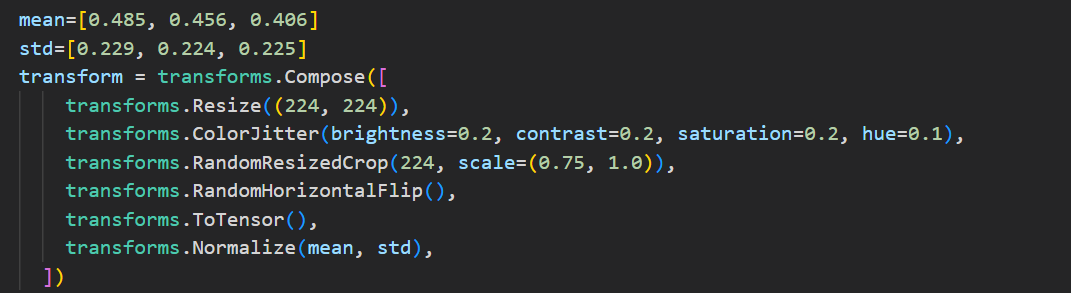


* 一張含有 文字 的圖片

  自動產生的描述一張含有 文字 的圖片

  自動產生的描述這邊**光是一個Epoch就要跑2.5小時**，所以決定只開5個Epochs
* 在這之前我嘗試了很多參數的設定，怎麼樣都是這種又臭又長且不合理的output，所以**推測是Epoch不夠**，因此我還是乖乖地拿出錢包儲值了Colab Pro

## Garbage Output\_2

* Data Augmentation
* 在更之前的作業，我曾經有在 “Data Augmentation” 加入 “**ColorJitter**”使Loss大幅下降過，又我看了網路上其他人的做法，當Loss降不下來，參數調來調去都是差不多的結果時也會加入“ColorJitter”，因此我在這次也決定加入參數調整
* Encoder Model & Decoder Model
* Pretrained model採用跟先前一樣的**Resnet152**
* 一張含有 文字 的圖片

  自動產生的描述一張含有 文字 的圖片

  自動產生的描述Training Settings
* 因為前面推測是Epoch 數不夠，所以我在train時設到**60 epochs**，並加入scheduler使learning rate每10 step就下降10倍，但在我train到第10小時(第50 epoch)的時後它自行中斷了☹，我看了一下Loss有出現0.9開頭了，但是實在是跑太久又怕失敗，所以我再次train就設**40 epochs**，總共耗費了7小時
* Training Result
* 經過了漫長的等待，看到Loss下降還興奮了一下，但看起來還是沒train成功☹ (這邊只截圖前後各10 epochs的Loss)

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

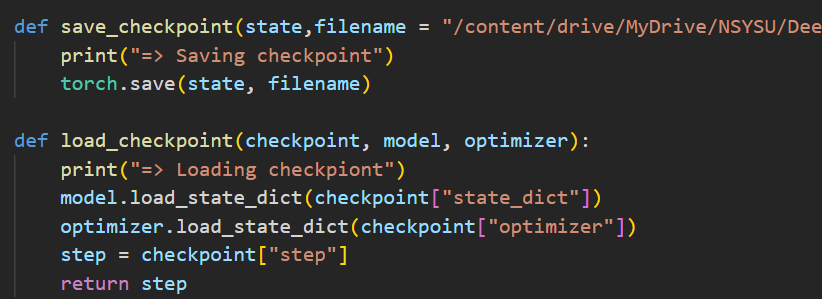
## Excellent Output

* 一張含有 文字 的圖片

  自動產生的描述Data Augmentation
* 跟第二個版本一樣，有加入**ColorJitter**來增強
* 一張含有 文字 的圖片

  自動產生的描述Encoder Model
* 因為先前train了多次的Resnet152，效果並沒有見好，所以決定改使用**Resnext101\_32x8d**
* 一張含有 文字 的圖片

  自動產生的描述一張含有 文字 的圖片

  自動產生的描述Training Settings
* 有了上一個版本的loss結果，這次就只train **20 epochs**，並且將**scheduler 的 step\_size 調為5**
* 比較特別的是因為怕模型在train的過程中又突然中斷，我有使用**checkpoints**來記錄每一個epoch的loss，以利後面中斷可以接續著train，就不用一直盯著模型看

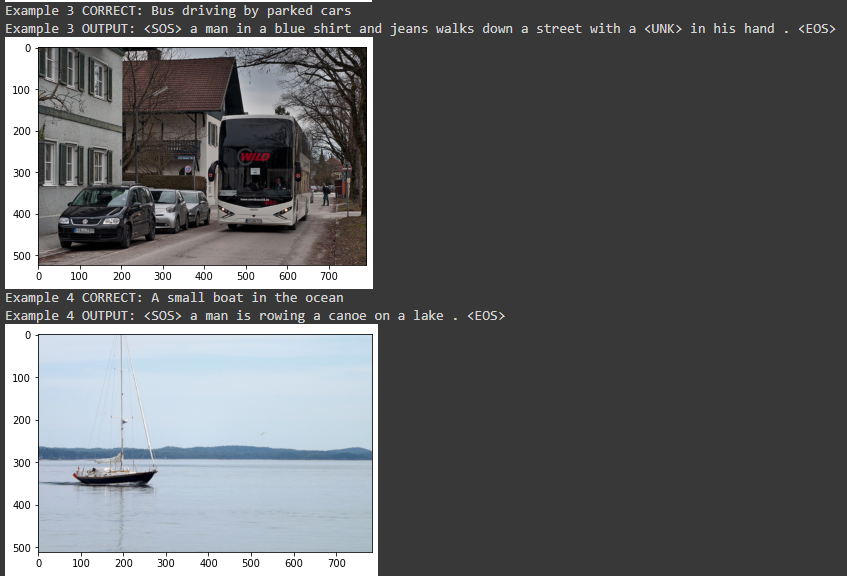
一張含有 文字 的圖片

自動產生的描述

* Training Result
* 一張含有 文字 的圖片

  自動產生的描述這次因為epoch數量減半，模型也有改變，總共耗費了3個多小時
* 一張含有 文字 的圖片

  自動產生的描述Loss最高在第3 epoch：2.7931，Loss最低在第11 epoch：1.2894

一張含有 文字, 螢幕擷取畫面, 電子用品, 顯示 的圖片

自動產生的描述

* 看起來是有train起來啦😊，終於~

Assignment 6 Part 2: Image Captioning with Attention

* 一張含有 文字 的圖片

  自動產生的描述Data Augmentation
* Resize：256
* CenterCrop：224
* 其他調整的功能除了使用RandomHorizontalFlip外還有加入**ColorJitter** 去對圖片進行調整，使data更加複雜
* Encoder Model

\_\_init\_()

一張含有 文字 的圖片

自動產生的描述

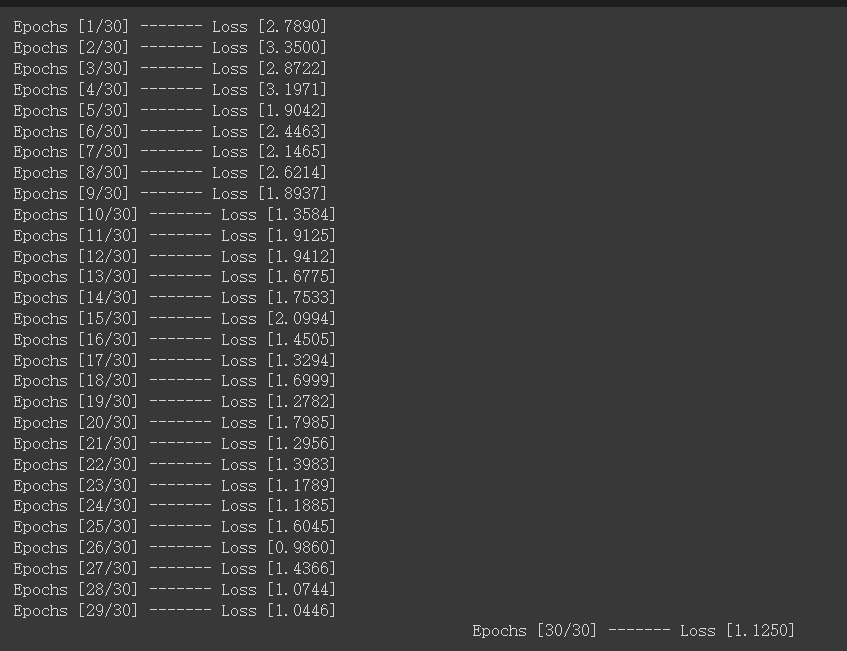
forward()一張含有 文字 的圖片

自動產生的描述

* 這次使用的pretrained model是**Resnet101**，選擇用這個model是因為我在網路上查相關資料時，大多都是看到使用Resnet，又使用Resnet101的範本最多，因此我也選擇這個model
* 一張含有 文字 的圖片

  自動產生的描述Training Hyperparams Setting
* 以上參數也是我參考網路上作法再自己邊train邊調的結果
* Training

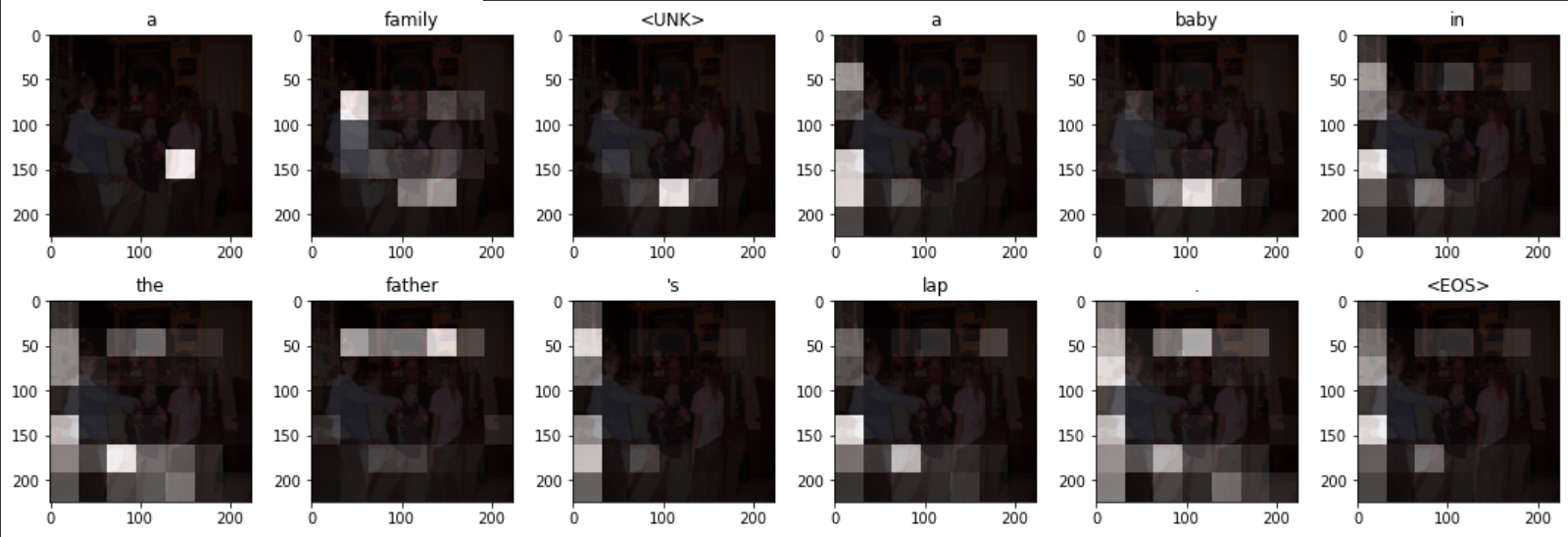
一張含有 文字, 時鐘 的圖片

自動產生的描述

* 起初一個epoch都要跑**將近3小時!!** 而升級了Colab Pro後一epoch只需要13分鐘左右
* 一開始我都是開5、10 epochs在跑，但train出來的結果很差，所以我看了一下所剩的colab pro單元，估算後決定開到30 epoch
* **30 epochs**總共花了6個多小時在train，所幸**Loss有從2.7降到1.0**

(Loss最高在第2 Epoch：3.3500，Loss最低在第26 Epoch：0.9860)

* Test Result



* 我個人覺得最後是有train起來的，如果我還有更多單元可用，我想我會再往上增加epoch的數量，來嘗試讓loss降到1.0以下