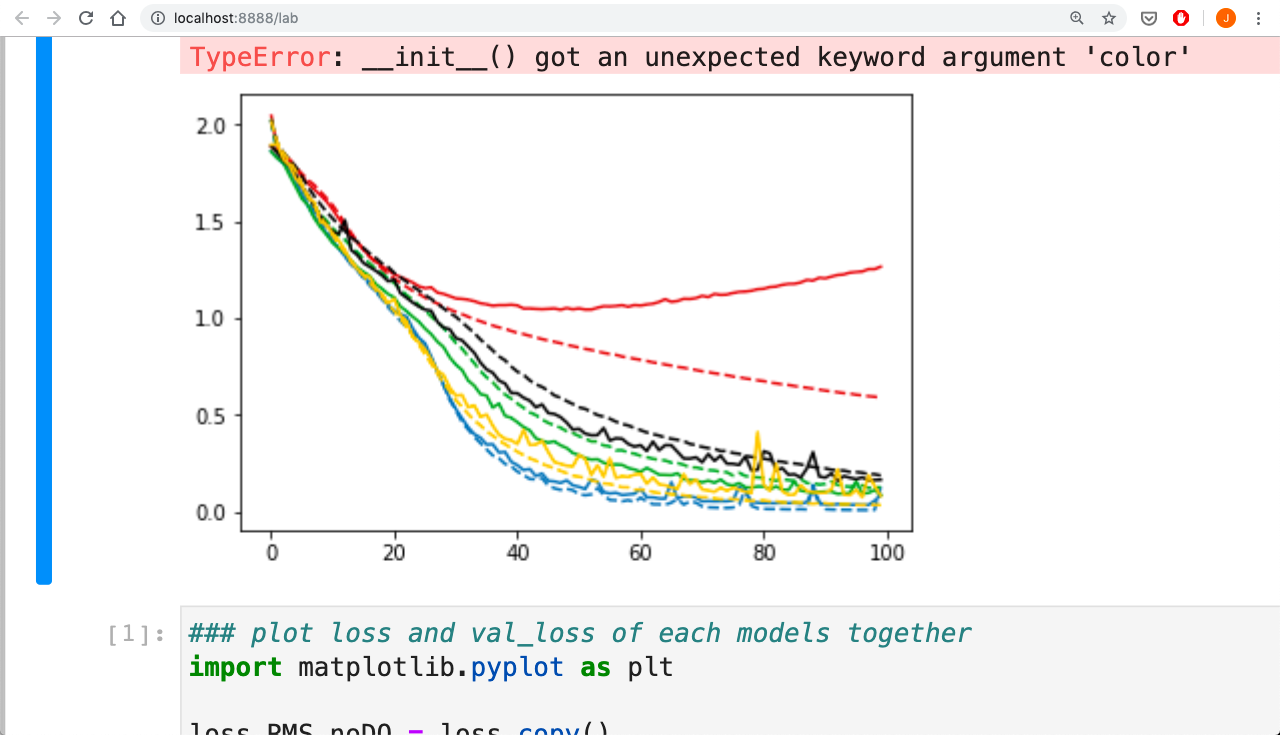
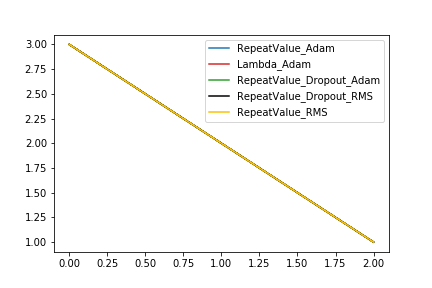
DSAI Assignment 2

1. Adder

Addition





loss

epoch

1. RepeatVector vs Lambda layer (紅色vs.藍色)
2. 說明：

我的model主要有三大層，依序為LSTM、LSTM、Dense。其中紅色表「第一層LSTM的return\_sequences=True，接著是一層Lambda(lambda x: x[:, -(DIGITS + 1):, :])，即只取第一層LSTM的最後DIGITS + 1個output vector」。藍色則表「第一層LSTM的return\_sequences=False，接著是一層layers.RepeatVector(DIGITS + 1)」。由上圖可見，在兩層LSTM間插入RepeatVector遠好於前者。

1. 原因猜想：

就物理意義而言，第一層LSTM是在encode（總結、理解）input的資訊，而第二層LSTM是在做運算、decode。而藍色model就相當於還沒看完整串input的算式，就開始總結資訊給下一層。至少在加減法問題，沒看完就開始總結資訊可能較不合理。比如說input算式是123+456，但encoder在看完123+45時，就告訴decoder說自己看到了123+45，然而123+45和123+456之間有頗大的差異（如：4從十位變百位），這樣123+45就成了雜訊。

1. Dropout（綠色vs藍色 或 黑色vs黃色）

無Dropout的model training loss下降得比較有Dropout的model快，很合理。同時，無Dropout的model的validation loss也同步下降，且在後期穩定的維持在很低的值，可能是因為加法問題不太容易overfit。

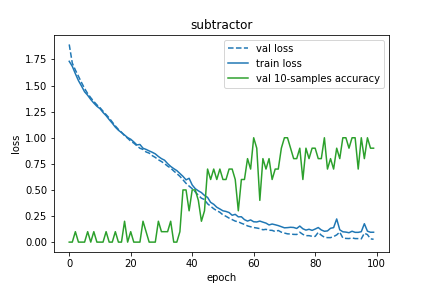
另外，有Dropout的model在訓練過程中，validation loss比training loss更低。應是model.fit()在training時有dropout，而在validation時沒有dropout所致。

1. Optimizer: RMSprop vs Adam（黃色vs藍色 或 黑色vs綠色）

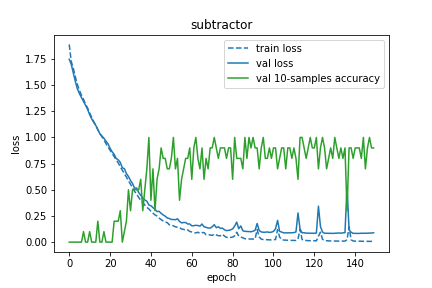
不論無dropout或有dropout，Adam都能訓練得比RMSprop快，可見Adam發揮momentum的性質，較不易在gradient descent時走彎路。

不論無dropout或有dropout，Adam的training曲線都能比RMSprop的training曲線隆起處來得少。

1. 根據以上，adder在testing時決定採用RepeatValue, 無dropout, Adam這些超參數訓練出的model。另外，亦決定以此設定開始subtractor和Mix的測試。
2. 將test\_x對應的prediction和test\_y進行比較（程式碼與在訓練時的每個epoch下，測試10 個validation sample的程式碼類似），得到加法運算正確率為94.6%。
3. Subtractor



此model的hyperparameter和「一、adder」中藍色線對應的model相同，但在adder，於epoch為100時，validation loss早已趨向穩定，而此subtractor在此時仍有一些下降趨勢（減法訓練得比加法來得慢，猜想是因為減法的借位等運算較為複雜），因此決定再以更多epoch (150)進行training和validate，並畫出下圖。

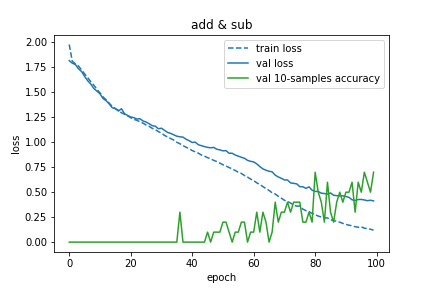


觀察到在loss趨穩後會有明顯的隆起。但之後又會回到原本的低loss rate，且training loss也是一起增加，所以應非所謂的overfit。目前不確定原因，只能猜想是loss space的特性。

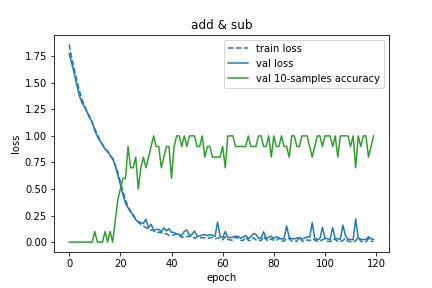
Loss再過epoch 100後確實有繼續降低，又因上圖的model最終落回了低loss，因此決定以上圖的model進行test。

將test\_x對應的prediction和test\_y進行比較（程式碼與在訓練時的每個epoch下，測試10 個validation sample的程式碼類似），得到減法運算正確率為90.8%。

1. Mix Task (add & subtract)



上圖的為training set = 18000（同前述的adder、subtractor）。而我們可見在後段有overfitting之情況。因此以下以training set = 54000進行training和validate，並畫出下圖。



Overfit的情況消失了，可見該training set能夠提供足夠的有效資訊。

以此model進行test。將test\_x對應的prediction和test\_y進行比較（程式碼與在訓練時的每個epoch下，測試10個 validation sample的程式碼類似），得到加/減法運算正確率為96.9%。

1. 乘法

我認為以相同的模型設定去訓練做乘法，也會有不錯的效果。因為乘法運算與加法運算能相互對應如下：

* 乘法的1\*1 = 1, 1\*2 = 2等九九乘法規則，對應到1+1 = 2, 1+2 = 3…等加法規則。
* 3\*9 = 27中，模型必須要識別出2是進位，對應到加法中的7+9 = 16模型可識別出1是進位。
* 乘法必須要把進位「加」到下一位的乘法運算結果上。這部分用的正是加法。