

# 4995 Deep Learning HW2

## Home Work #2

va2361: Vishal Anand, ss5348: Somya Singhal

March 31, 2018

### Question 1: CIFAR-10

1.1 Accuracy **70.32%**

1.2 Accuracy **77.73%**

#### 1.1

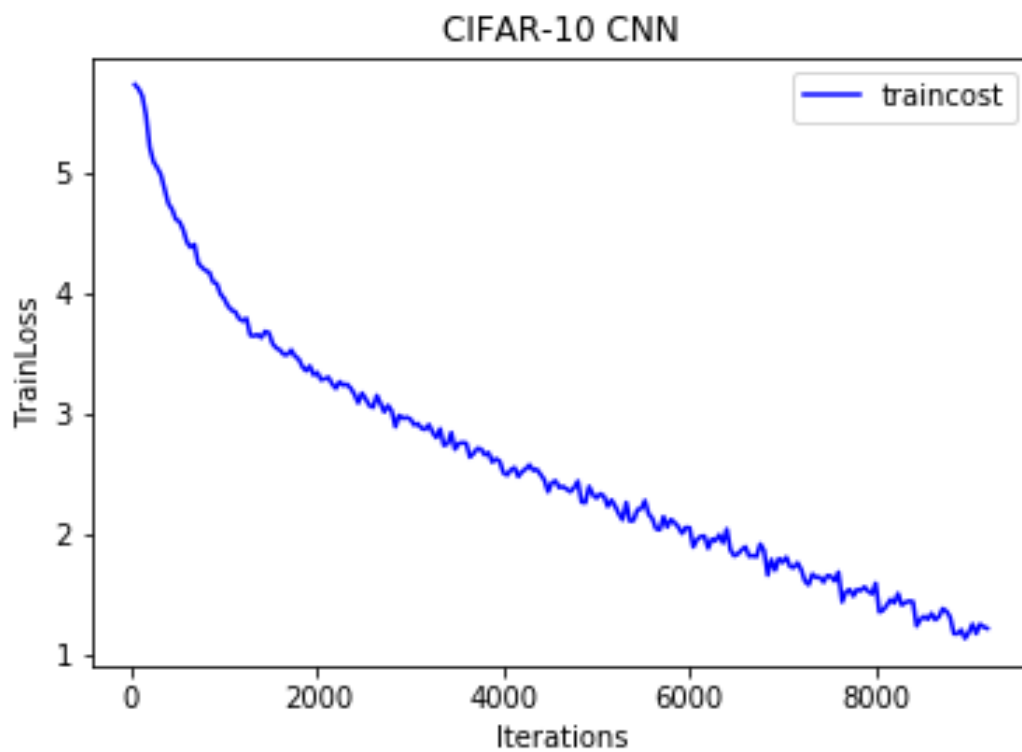
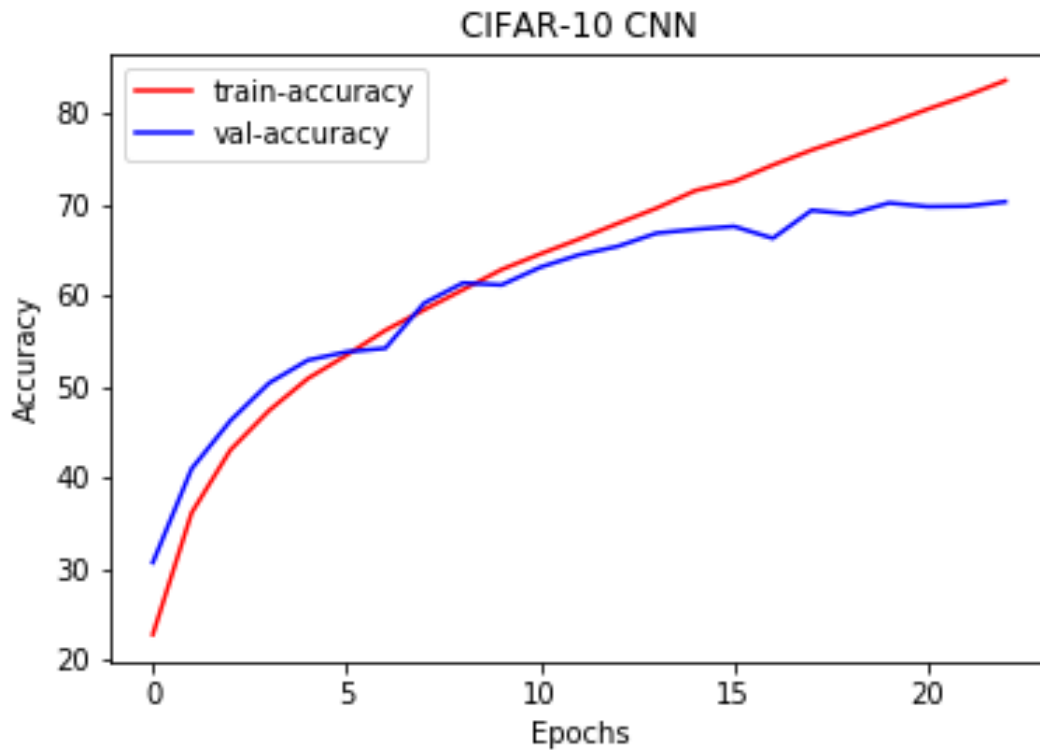
##### Architecture

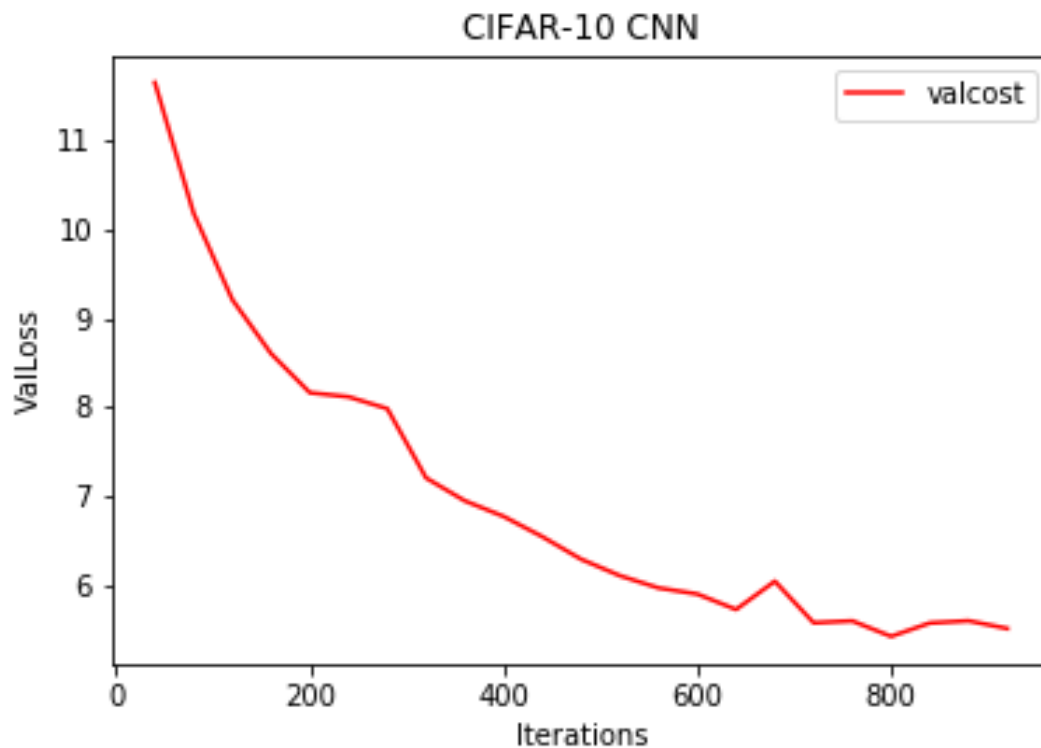
Convolution(3,64,5) - in-channel = 3, out-channel = 64, kernel = 5\*5, stride = 1\*1  
Relu()  
MaxPool(2,2)-max pool window-size = 2\*2  
Convolution(64,256,5) - in-channel = 64, out-channel = 256, kernel = 5\*5, stride = 1\*1  
Relu()  
MaxPool(2,2)-max pool window-size = 2\*2  
view(-1, 256 \* 5 \* 5)  
Fully Connected(256 \* 5 \* 5, 400)-in-channels = 256\*5\*5, out-channels = 400  
Relu()  
Fully Connected(400,10)- input-channels = 400, out-channels = 10

Loss function used is : CrossEntropyLoss  
Optimizer used is: SGD  
Learning rate =0.01  
epochCount = 23  
batchSize = 40

Final train accuracy = 83.64271436216889  
Final Validation accuracy = 70.32388663967612  
Final Test accuracy = 70.32

## Graphs





## 1.2

### Delta from 1.1

- Modified the architecture of the CNN
- Used dropout
- Used different optimizer
- Used different learning-rate

### Architecture

Convolution(3, 64, 3)- input-channel = 3, out-channel = 64, kernel = 3\*3, stride = 1\*1  
Relu()

Convolution(64, 128, 3)- input-channel = 64, out-channel = 128, kernel = 3\*3, stride = 1\*1  
Relu()

MaxPool(2,2)- window-size = 2\*2

Dropout(0.25)

Convolution(128, 256, 3)- input-channel = 128, out-channel = 256, kernel = 3\*3, stride = 1\*1

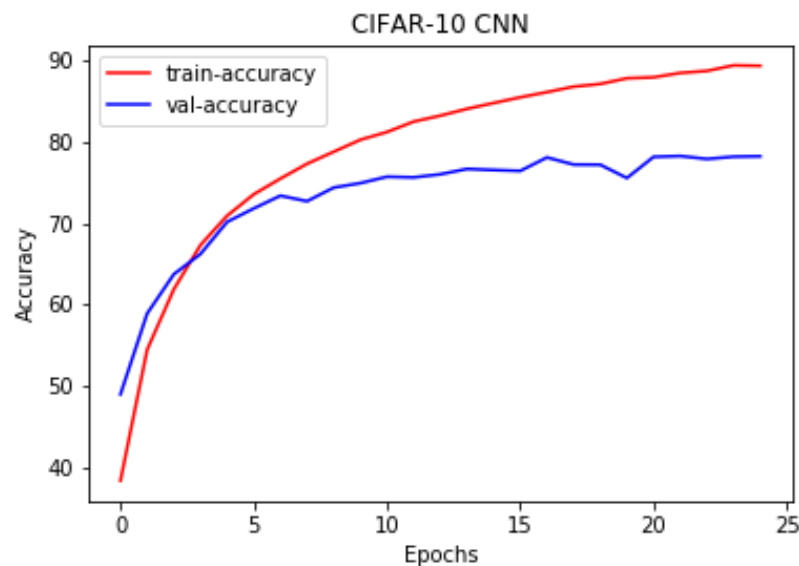
Relu()

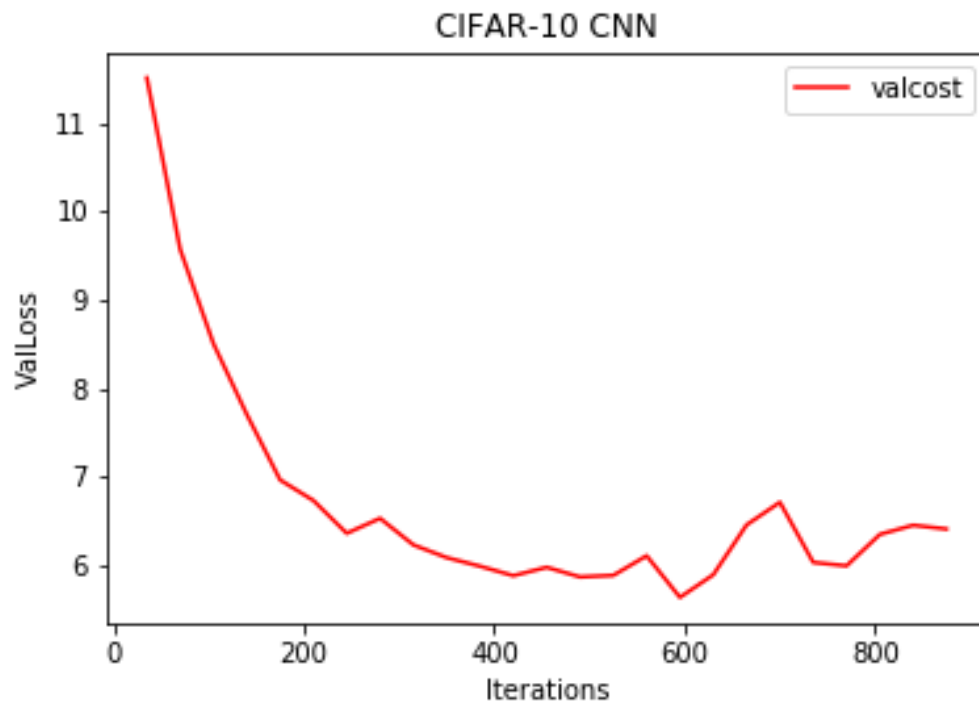
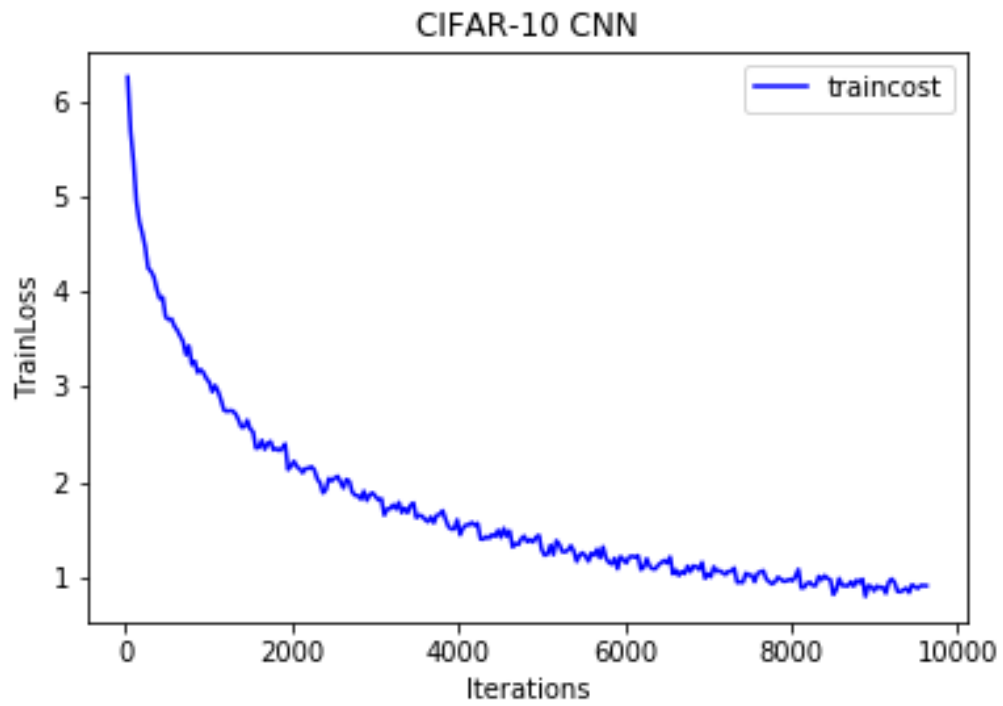
Convolution(256,256,3)- input-channel = 256, out-channel = 256, kernel = 3\*3, stride = 1\*1  
Relu()  
MaxPool(2,2)- window-size = 2\*2  
Dropout(0.25)  
view(-1, 256 \* 5 \* 5)  
Fully Connected(256 \* 5 \* 5, 512)- input-channel = 256\*5\*5, out-channel = 512  
Relu()  
Dropout(0.25)  
Fully Connected(512, 10)- input-channel =512, out-channel = 10

Loss function used is: CrossEntropyLoss  
Optimizer used is: RMSprop  
Learning rate = 0.0003  
epochCount = 25  
batchSize = 35

Final train accuracy = 89.32014501234471  
Final Validation accuracy = 78.21801891488303  
Final Test accuracy = 77.734335839599

## Graphs



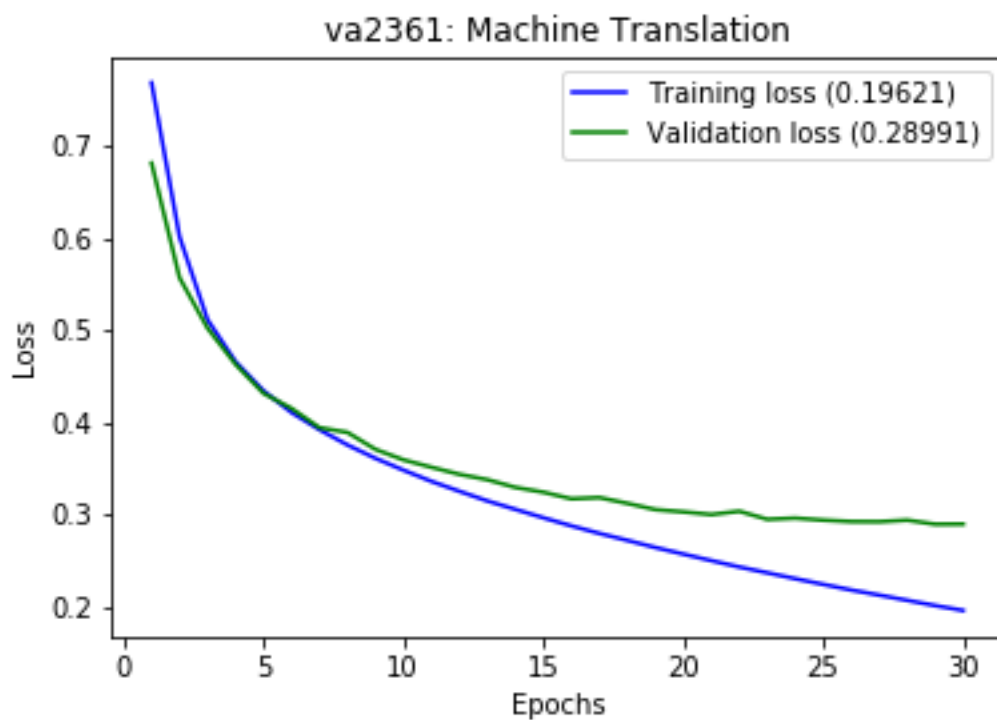


## Question 2: Machine Translation

BLEU-Type	Value
Mean Sentence-BLEU	0.4976887
Corpus-BLEU	0.0514969

Table 1: Test-set evaluation

### Graphs



### Translations

Algorithms used frameworks [1] and [2]. All the predicted translations are located in the github link. Code is attached with the zip file, and the same is also at <https://github.com/vishalanand/DL-4995-LSTM-MT>

Best translations		
BLEU	Prediction	Actual
1.0	Kannst du das machen?	Kannst du das machen?
1.0	Ist Tom da?	Ist Tom da?
0.903602	Enspann dich bitte.	Bitte setz dich!
0.88011174	Das kann ich nicht sagen.	Ich kann nicht schuldern.
0.88011174	Lassen Sie das mich machen.	Lass mich das machen.
0.84089642	Das wird nicht gehen.	Das war nicht gesterben.
0.84089642	Zeig mir das mal!	Zeig es mir.
0.84089642	Tom mag mich nicht.	Tom hat mich angesucht.
0.84089642	Du kannst nicht verlieren.	Du bist nicht hier.
0.84089642	Hilf Tom!	Hilfe Tom.
0.84089642	Ist es nicht eben?	Ist das nicht Tom?
0.84089642	Sie findet ihn hinreiend.	Sie hat ihn gefunden.
0.84089642	Die Luft ist feucht.	Die Tasse ist gest.
0.84089642	Untersttze Tom!	Hilf Tom.
0.84089642	Tom war nicht nass.	Tom wird nicht hertig.
0.84089642	Tom sah ruhig aus.	Tom sieht gut aus.
0.84089642	Wir knnen Tom helfen.	Wir haben Tom gefunden.
0.84089642	Tom rhrte sich nicht.	Tom hat nicht gestorben.
0.84089642	Sie kann nicht schwimmen.	Sie wird nicht gehen.
0.84089642	Wie breit ist es?	Wie geht es dir?
0.84089642	Hallo, Mdels!	Hallo, Tom.
0.84089642	Wie gewagt!	Wie gerabendeit!
0.84089642	Tom sah blass aus.	Tom sieht gut aus.
0.84089642	Du hast mich angelogen.	Du wirst mich beschftigt.
0.84089642	Ich wurde nicht genommen.	Ich war nicht schwirren.
0.79527073	Tom geht es nicht gut.	Tom ist nicht schwimmen.
0.79527073	Das kann gar nicht sein.	Das war nicht gest.
0.75983569	Wer hat aufgehrt?	Wer ist das?
0.75983569	Tom war schmutzig.	Tom wird gehen.
0.75983569	Tom hat angerufen.	Tom ging geschen.

Table 2: Sorted 30 best predicted translation alongside expected translations

## Model description

**Hyper-parameters:** batch\_size = 40, latent\_dim = 250, epochs = 30

Model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None, 69)	0	
input_2 (InputLayer)	(None, None, 85)	0	
lstm_1 (LSTM)	[(None, 250), (None, 320000		input_1[0][0]
lstm_2 (LSTM)	[(None, None, 250), 336000		input_2[0][0] lstm_1[0][1] lstm_1[0][2]
dense_1 (Dense)	(None, None, 85)	21335	lstm_2[0][0]
Total params: 677,335			
Trainable params: 677,335			
Non-trainable params: 0			

Encoder model

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, None, 69)	0
lstm_1 (LSTM)	[(None, 250), (None, 250)	320000
Total params: 320,000		
Trainable params: 320,000		
Non-trainable params: 0		

Decoder model

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, None, 85)	0	
input_3 (InputLayer)	(None, 250)	0	
input_4 (InputLayer)	(None, 250)	0	
lstm_2 (LSTM)	[(None, None, 250), 336000		input_2[0][0] input_3[0][0] input_4[0][0]
dense_1 (Dense)	(None, None, 85)	21335	lstm_2[1][0]
Total params: 357,335			
Trainable params: 357,335			
Non-trainable params: 0			



## References

- [1] Chollet, François et al. *Keras*  
<https://github.com/keras-team/keras>
- [2] *PyTorch*  
<http://pytorch.org/>