Fast Intent Classification for Smart Assistants

Akshit Tyagi, Varun Sharma, Lynn Samson, Nan Zhuang, Zihang Wang
UMass, Amherst with Amazon

Abstract

Smart voice assistants have started to become an integral part of our daily interaction. The assistant needs to be quick in inferring the correct response with a certain degree of confidence. However, these assistants generally lack the flexibility to reason deeper on specific requests. Distinguishing which requests need more inference and which ones can be handled by shallow thinking makes the responses of the assistant seem richer and more human-like. Recently, early exiting strategies have been used successfully for deep convolutional networks. We propose a strategy that combines these early exiting strategies with conventional NLP networks, making the agent learn when it can predict from a shallower branch and when it needs to go through more layers of the same network to respond more confidently. Overall we see improvements in computational load of the network

Problem

- **Natural Language Understanding:** The task of intent classification takes in a query x_i (is a D dimensional embedding from a bag-of-words model, and an $L \cdot D$ embedding for a sequential model, where L is the maximum sequence length that we consider), and maps it to a label y_i which is one of C intent classes.

- **Fast Inference:** The second part of the problem involves making inference for the intent classifier, faster. We have modelled this problem as minimizing the number of FLOPS required to infer the class for a single sample, as well as the average FLOPS for the entire test data set.

Datasets

- **ATIS:** realistic conversation which contains corrections and colloquial expressions.
- **FSPS:** crowd sourced to provide what people would ask a system that could assist in navigation and event querying

Corpus	Total	Train	Dev	Test
ATIS	5871	4478	500	893
FSPS	44783	31279	4462	9042

Table 1. Dataset splits

Models

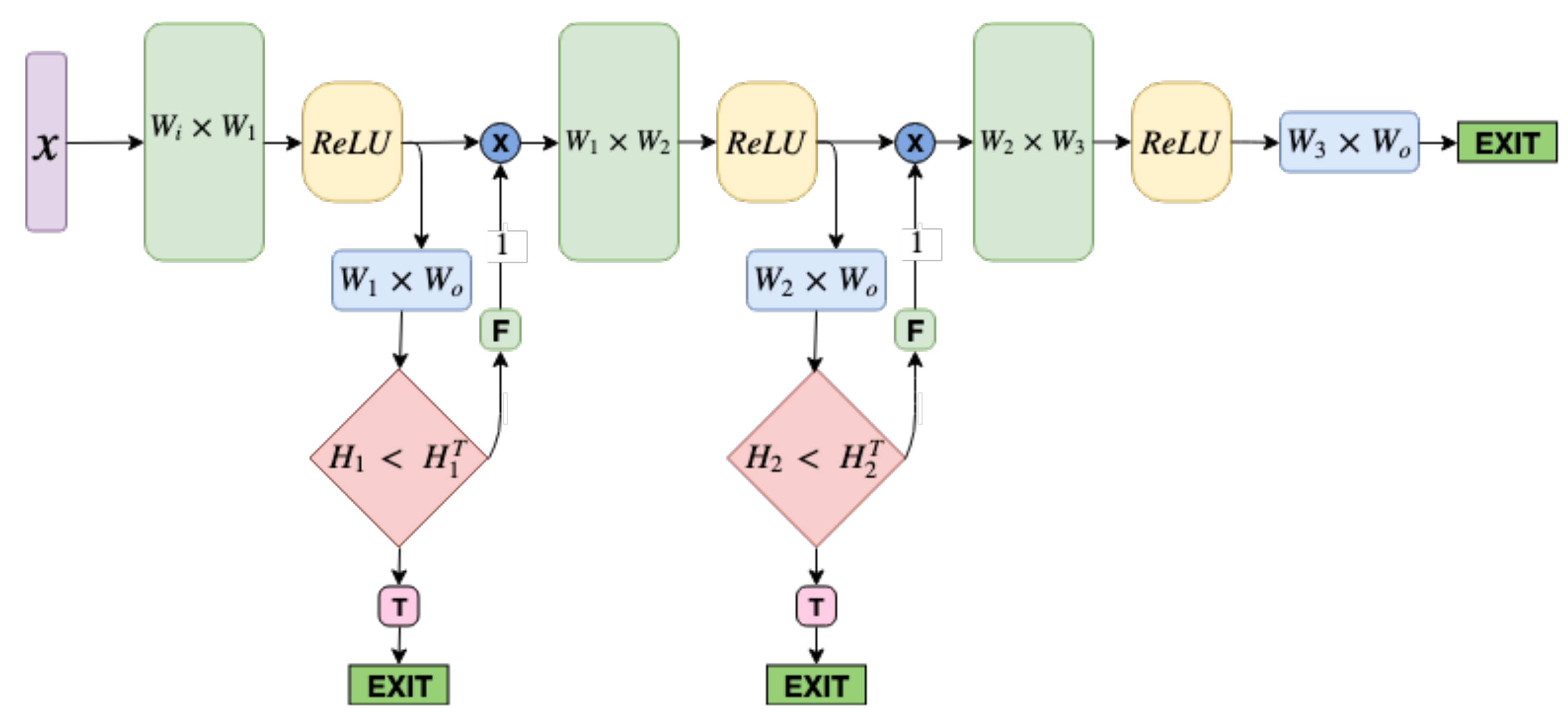


Figure 1. Feed Forward Net with Early Exiting

Exit Point	# Params($\times 10^3$)	FLOPS($\times 10^3$)
3-Layer Regular Net	**36.4**	**36.2**
3-LayerEarlyExit-e_1	32.6	32.5
3-LayerEarlyExit-e_2	38.9	38.7
3-LayerEarlyExit-e_3	40.8	40.6

Table 2. Computational load variation for a FeedForward net

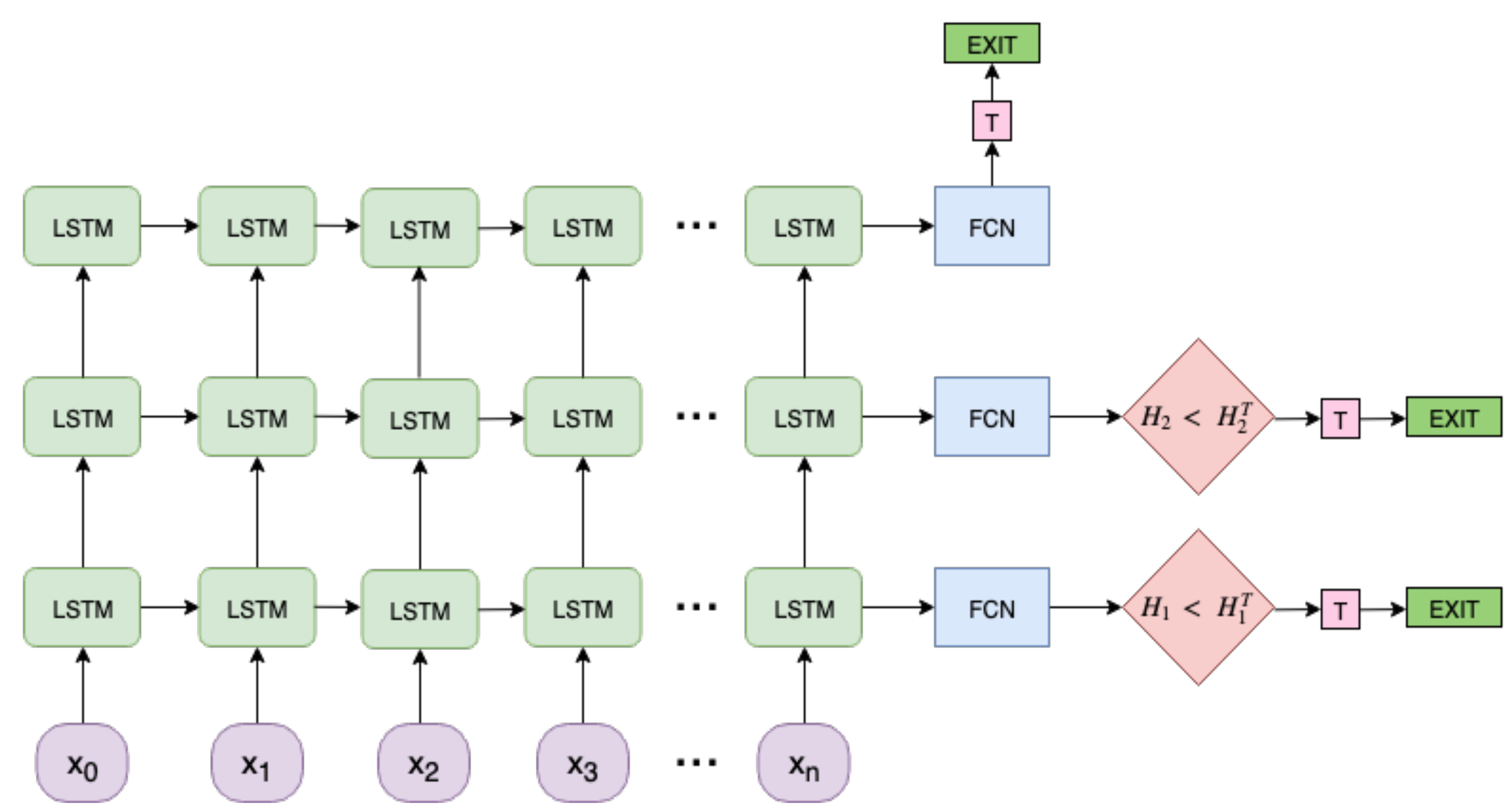


Figure 2. Stacked LSTM with Early Exiting

Exit Point	# Params($\times 10^3$)	FLOPS($\times 10^3$)
Stacked-LSTM	**22,004**	**69,192**
Stacked-LSTM-e_1	7,655	23,084
Stacked-LSTM-e_2	14,860	46,143
Stacked-LSTM-e_3	22,065	69,202

Table 3. Computational load variation for a Stacked-LSTM

Results

Dataset	F1	Precision	Recall	Accuracy
FSPS	0.29	0.41	0.28	0.80
ATIS	0.19	0.34	0.19	0.79

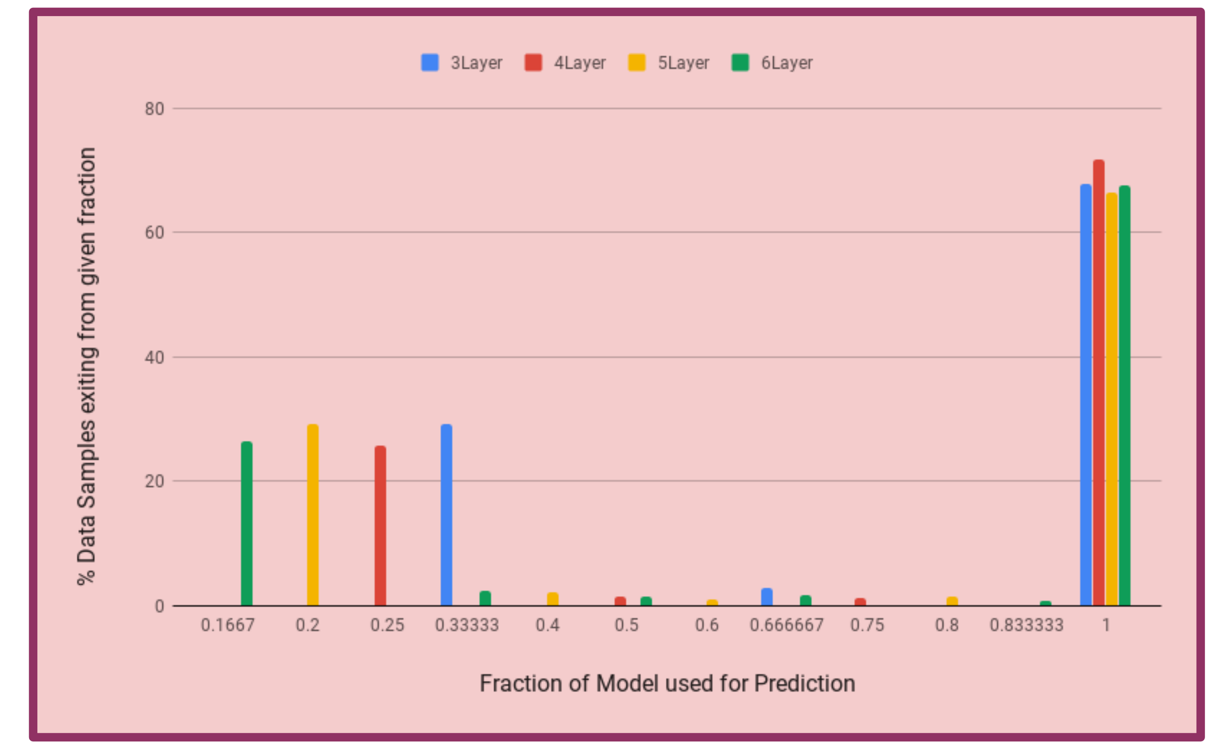
Table 4. Baseline: Naïve Bayes

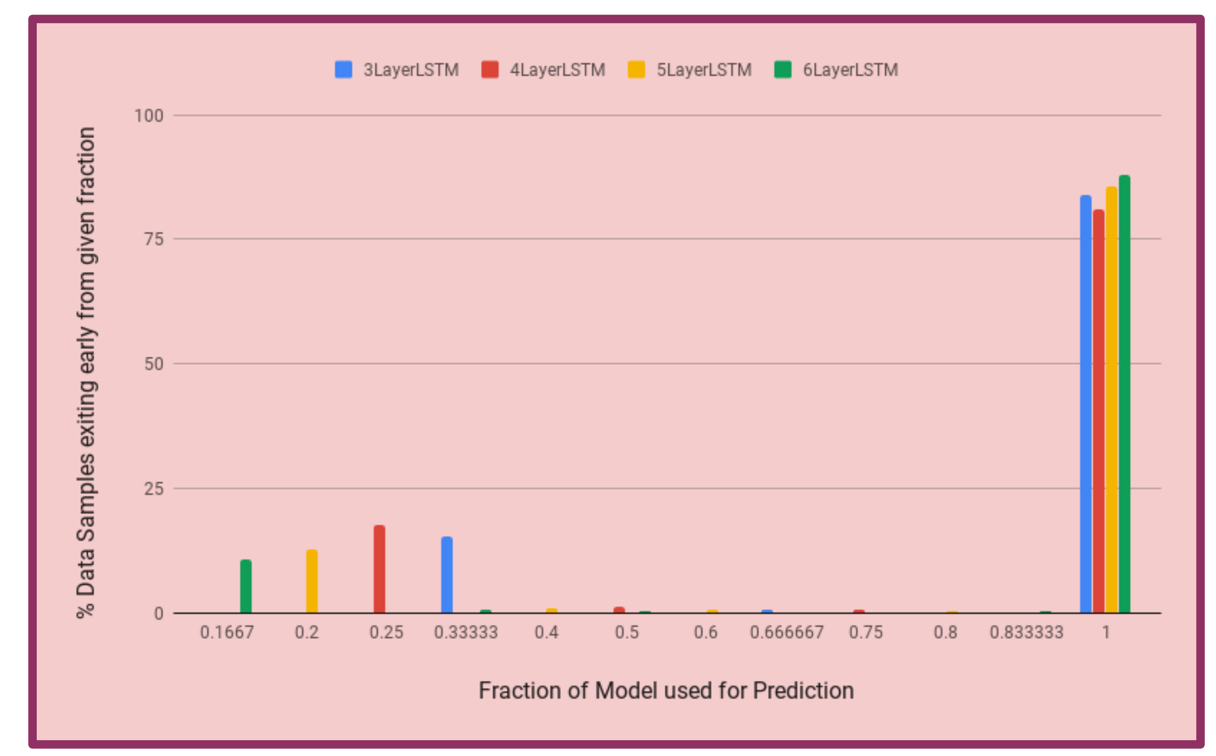
Model	F1(Macro)	Acc.(%)
ThreeLayer	0.48	88.5
ThreeLayerEarlyExit	0.55	89.6
S-LSTM	0.65	92.8
S-LSTMEarlyExit	0.66	93.2

Table 5. Early Exit v/s Vanilla Networks

Accuracy(%)	Exit Points[e_i :% exit]
85.03	1: **29.30** 2: 02.60 3: 67.90
87.84	1: 27.37 2: 01.80 3: **70.80**
89.20	1: 27.80 2: 01.92 3: 70.20

Table 6. Variation of Exit % with Accuracy





Conclusions

- The new weighted average loss function does a good job of regularizing the overall network
- Most of the predictions are either from the first layer or the last layer of the network
- As accuracy improves more and more predictions exit from the last layer, as expected
- Keeping the first layer of the network on the device can allow for a ~30% reduction on server load while maintaining model performance near the state of the art