Model Compression of Sequential Networks

Oshin Dutta
Ayush Srivastava

Contents

Introduction

- LSTMs and GRUs - huge number of parameters, hence training process notoriously difficult and easily over-fitting. For high dimensional inputs such as video frames, input-to-hidden matrix is extremely large.
- Learning the compact structures in Recurrent Neural Networks (RNNs) is more challenging. As a recurrent unit is shared across all the time steps in sequence, compressing the unit will aggressively affect all the steps. ISS[1] involves simultaneously decreasing the sizes of all basic structures in LSTM one by one, based on group Lasso regularization. It achieves about 3x reduction in parameters and 10x speedup.
- VIBnet[4] developed for CNN and FC layers utilizes the information bottleneck principle instantiated via a tractable variational bound. Minimization of this information theoretic bound reduces the redundancy between adjacent layers by aggregating useful information into a subset of neurons that can be preserved, the rest are shut off by the sparse nature of the framework.

Related Work

- [4] forms Tensor Ring-LSTM by utilizing the low-rank tensor ring decomposition (TRD) to reformulate the input-to-hidden transformation of input to LSTMs. They evaluate on UCF11 and HMDB51.
- In [5], comparison of tensor decomposition methods was done on polyphonic music music dataset, where Tensor Train-GRU performed the best. Further, on incorporating TT-GRU in Deepspeech2 and evaluating on librispeech, they compressed DS2 **350x** with **3%** increase in Valid. CER.
- [6] uses Krocker product based compression to achieve 16-38x compression of LSTM based architecture with MNIST, USPS digit recognition, KWS on Google command dataset, Human Movement Recognition on 3 publicly available datasets.
- [7] uses TT-decomposition for language modelling task and shows compression on Penn Tree Bank dataset.
- [8] develops Hybrid Matirx Decomposition, a variant of Low-rank Matrix factorisation that compresses RNNs by 2-4x with high inference speed, evaluated over KWS, HAR and PTB datasets.
- [9] compares semi-NMF, SVD and prunning techniques on PTB and Wiki-Text2 datasets for language modelling concluding that SVD works best. Stanford Question answering, Natural Lnaguage inference, sentiment treebank datasets were used for NLP task with pre-trained embeddings for language models- where pruning method worked best.

Variational Information Bottleneck Theory (VIB)

Variational Information Bottleneck[2]:

- Loss function per layer

$$\mathcal{L}_i = \gamma_i \boldsymbol{I}(\boldsymbol{h}_i; \boldsymbol{h}_{i-1}) - \boldsymbol{I}(\boldsymbol{h}_i; \boldsymbol{y})$$

- Variational bound

$$\tilde{\mathcal{L}}_i = \gamma_i \mathbb{E}_{\boldsymbol{h}_{i-1} \sim p(\boldsymbol{h}_{i-1})} [\mathbb{KL}[p(\boldsymbol{h}_i | \boldsymbol{h}_{i-1}) || q(\boldsymbol{h}_i)]] - \mathbb{E}_{\{\boldsymbol{x},\boldsymbol{y}\} \sim \mathcal{D}, \boldsymbol{h} \sim p(\boldsymbol{h}|\boldsymbol{x})} [\log q(\boldsymbol{y}|\boldsymbol{h}_L)] \geq \mathcal{L}_i$$

- Final Loss function

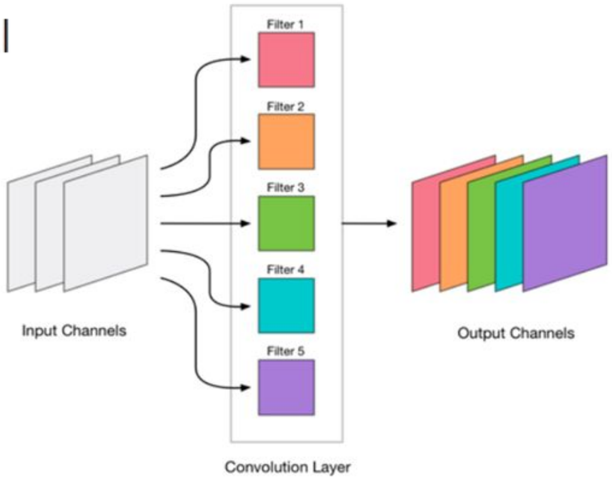
$$\tilde{\mathcal{L}} \triangleq \sum_i \tilde{\mathcal{L}}_i$$

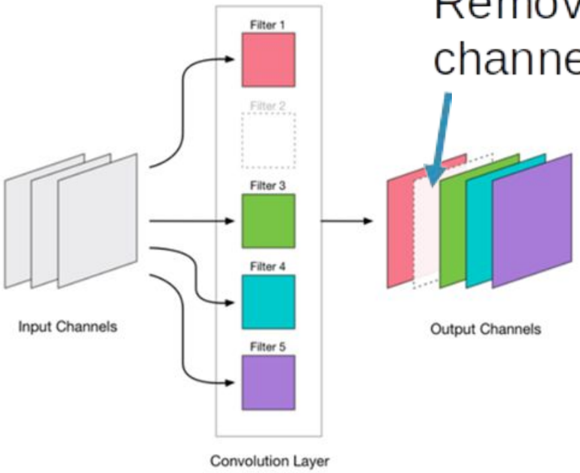
- With Gaussian assumptions

$$\tilde{\mathcal{L}} = \sum_{i=1}^{L} \gamma_i \sum_{j=1}^{r_i} \log \left(1 + \frac{\mu_{i,j}^2}{\sigma_{i,j}^2}\right) - L\, \mathbb{E}_{\{\boldsymbol{x},\boldsymbol{y}\} \sim \mathcal{D}, \boldsymbol{h} \sim p(\boldsymbol{h}|\boldsymbol{x})} [\log q(\boldsymbol{y}|\boldsymbol{h}_L)]$$

Compression with VIB

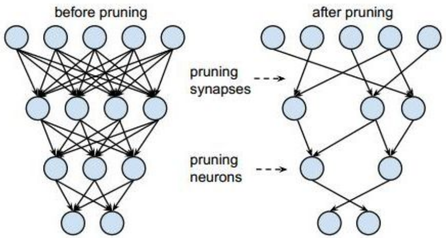
Convolutional architecture





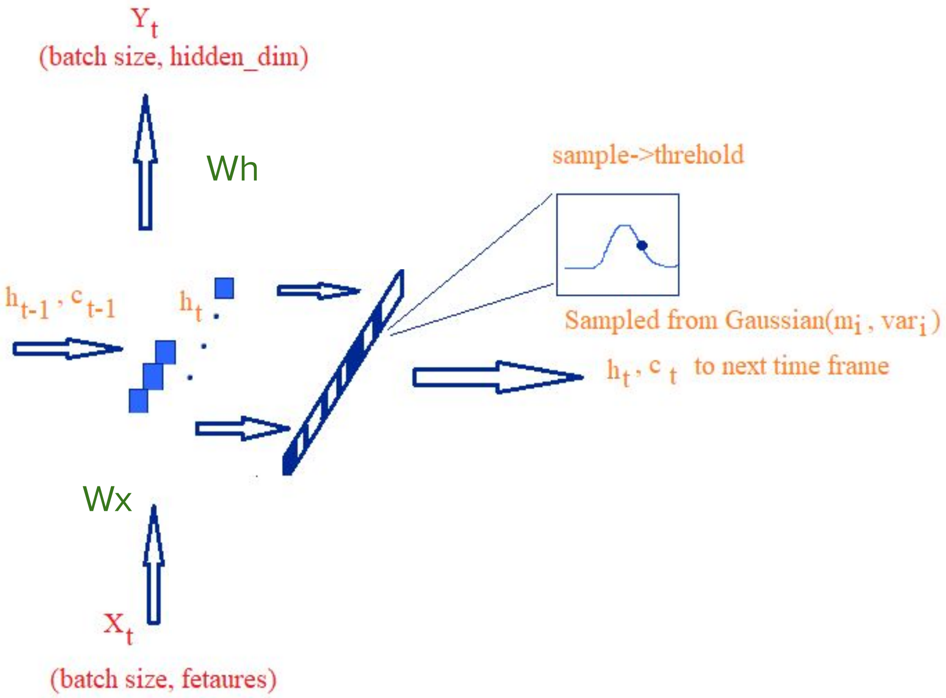
Removed channel

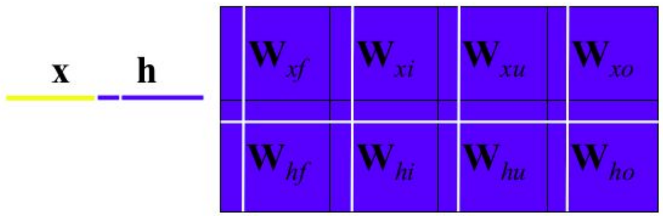
Fully Connected architecture



LSTM-VIB compression

- Inspired from VIB[4], we construct an algorithm which reduces the structure of LSTM, while preserving relevant information.
- Reduces input-hidden transformation matrix size
- Prunes out redundant hidden states, thus reducing overall size of all weight matrices in LSTM.





Weights matrices in LSTM

Compression Experimentation with LSTM-VIB

- Datasets used- UCF101 , UCF11
- Architecture tested on -
  - Convolutional- LSTM:
    - Feature extractors - pretrained resnet152- 58.14M , efficientnetb0- 13.38M
  - End-to-end LSTM
- Hardware specifications
  - NVIDIA K40 GPU
  - NVIDIA V100
- Amount of compression- Dataset and Architecture dependent

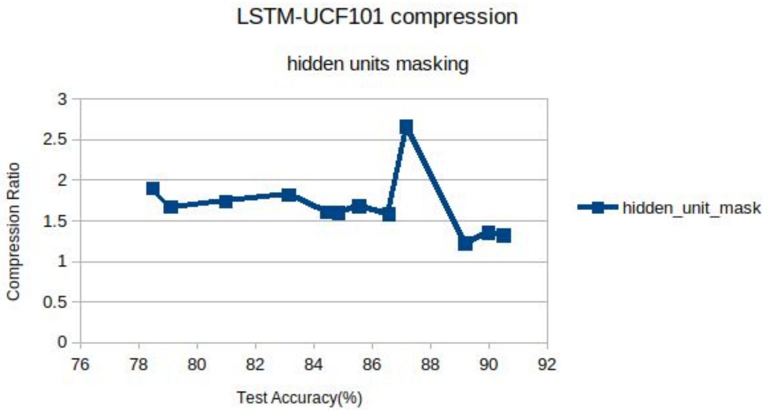
Compression of ConvLSTM-UCF101

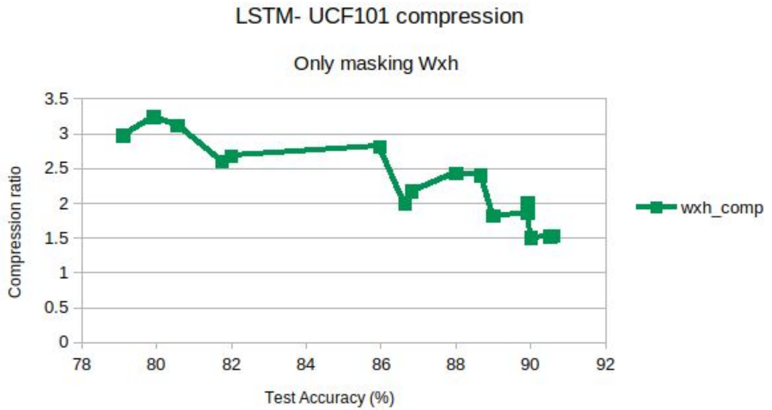
- Dataset- UCF101
  - 101 action classes- eye makeup, baby crawling, playing dhol, shaving, surfing haircut among others
- Uncompressed ConvLSTM -UCF101 : Top1 accuracy- 91%
- Original model size : 266 MB
- Weight parameters:
  - Feature extractor ~ 89%
  - LSTM ~ 9%
  - FC ~ 2%

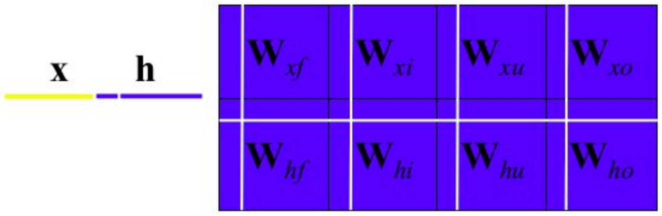
Compression of ConvLSTM-UCF101

Combinations tried out with different hyperparameters:

1. Hidden unit masking
2. Only Wx pruning -masking latent feature inputs





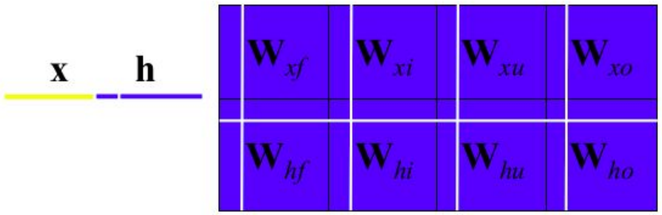


Weights matrices in LSTM

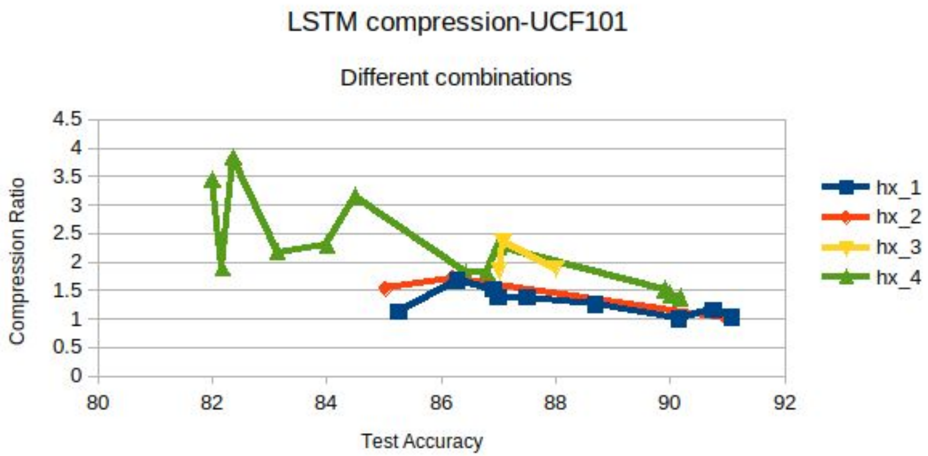
Compression of ConvLSTM-UCF101

Wh+ Wx pruning : masking gates outputs + input latent features

- Wh=Wx/2
- Wh=Wx/4
- Wh=Wx
- Wh=Wx*2



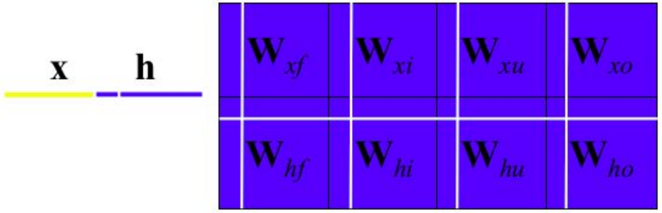
Weights matrices in LSTM



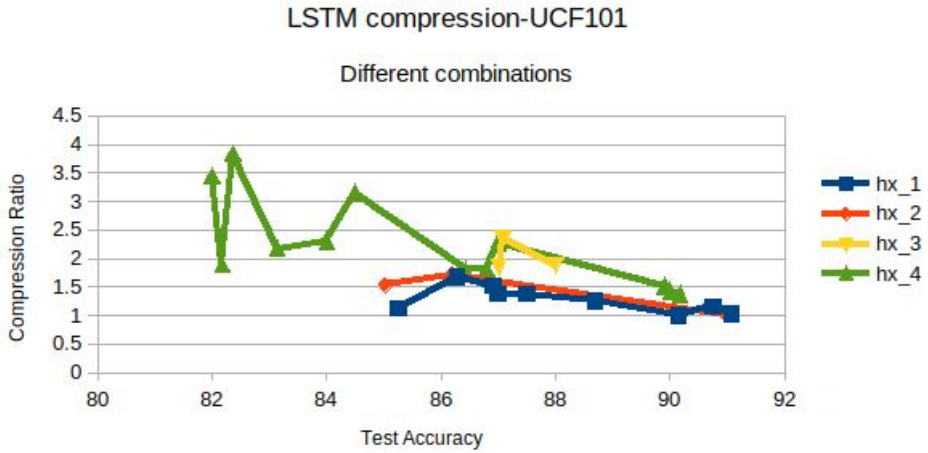
Compression of ConvLSTM-UCF101

Wh+ Wx pruning : masking gates outputs + input latent features

- Wh=Wx/2
- Wh=Wx/4
- Wh=Wx
- Wh=Wx*2



Weights matrices in LSTM



For the same accuracy, hidden states can be reduced at least twice as much as feature inputs to the LSTM

Compression of ConvLSTM-UCF101

- Optimum Test Accuracy-compression ratio :
    - 85% for 3.05x compression ie. 43.7% of original LSTM parameters remain (comparable to [1])

Test Accuracy	Compression Ratio
91%	1
87%	2.27
84%	3.14
82.36	3.83

Compression- UCF11-ConvLSTM

- UCF11 dataset
  - 11 action classes- basketball throw, diving, playing golf, tennis, juggling, walking dog among others
- Uncompressed ConvLSTM Model
  - Feature extractor resnet152 - 58.14M parameters o 233.4 MB

	Input x hidden state sizes	LSTM parameters	Valid. accuracy
Model 1	1024x2048	25.18M	**95.44%**
Model 2	256x512	1.58M	**97.10%**
Model 3 (pretrainedLSTM weights- UCF101)	512x1024	6.3M	**98.9%**

Our Accuracies

State of art on UCF11 as per [4]

Method	Accuracy
(Hasan and Roy-Chowdhury 2014)	54.5%
(Liu, Luo, and Shah 2009)	71.2%
(Ikizler-Cinbis and Sclaroff 2010)	75.2%
(Liu, Shyu, and Zhao 2013)	76.1%
(Sharma, Kiros, and Salakhutdinov 2015)	85.0%
(Wang et al. 2011)	84.2%
(Sharma, Kiros, and Salakhutdinov 2015)	84.9%
(Cho et al. 2014)	88.0%
(Gammulle et al. 2017)	**94.6%**
CNN + LSTM	92.3%
CNN + TR-LSTM	93.8%

Compression- UCF11-ConvLSTM

- Compression process-

  LSTM-VIB based mask training

  ⇩

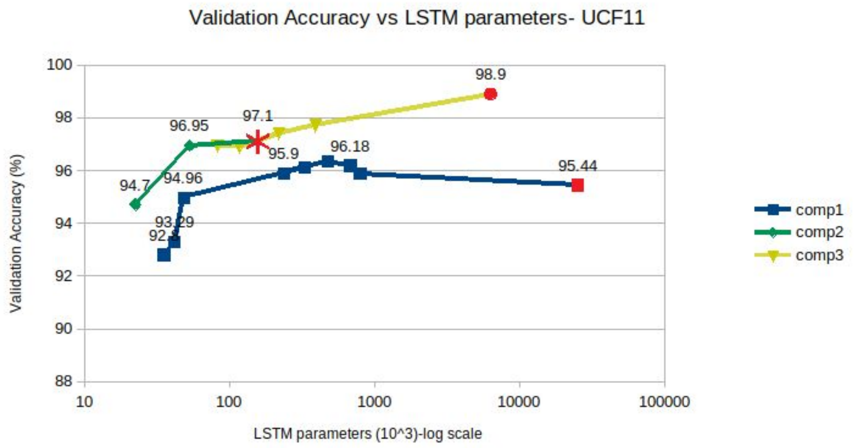
  Fine-tuning

  ⇩

  Exporting smaller model

- Compression Ratio- With ~1% accuracy degradation

Compressed	Compression Ratio	LSTM parameters	Valid. accuracy
Model1	520x	48k	94.3%
Model2	30x	53.4k	96.5%
Model3(pretrained with UCF101)	29x	219k	97.41%

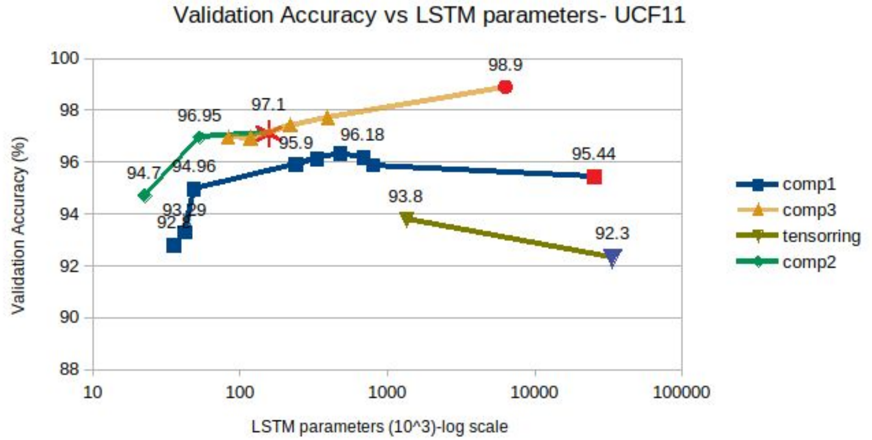


Red- uncompressed models
Other points- compressed models of uncompressed versions

Compression- UCF11-ConvLSTM

- **Comparison** with Tensor ring- UCF11[4]

Compression technique	Validation Accuracy	Parameters
Two-stream LSTM	94.6%	141M
TensorRing	93.8%	1.34M
Ours	93.29%	**0.041M**
	97.56%	**0.392M**



Red, purple- uncompressed models
Other points- compressed models of uncompressed versions

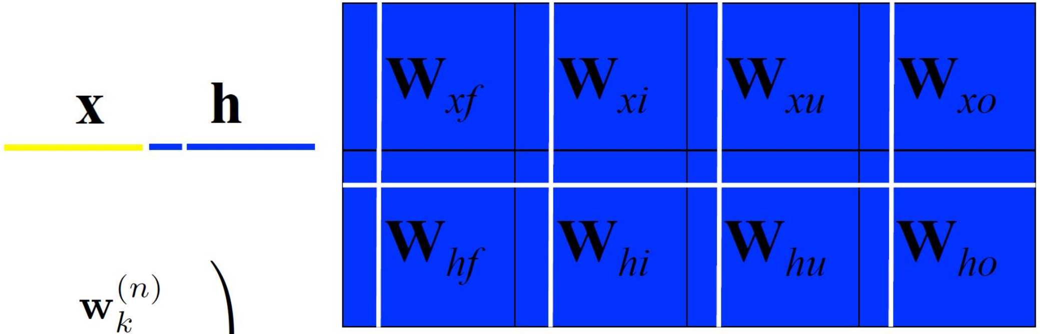
- Additionally, LSTM-VIB reduces about
  - 10x FC parameters

Intrinsic Sparse Structure (ISS) in LSTM

- A group lasso regularization.

$$R(\mathbf{w}) = \sum_{n=1}^{N} \sum_{k=1}^{K^{(n)}} \left|\left|\mathbf{w}_k^{(n)}\right|\right|_2$$

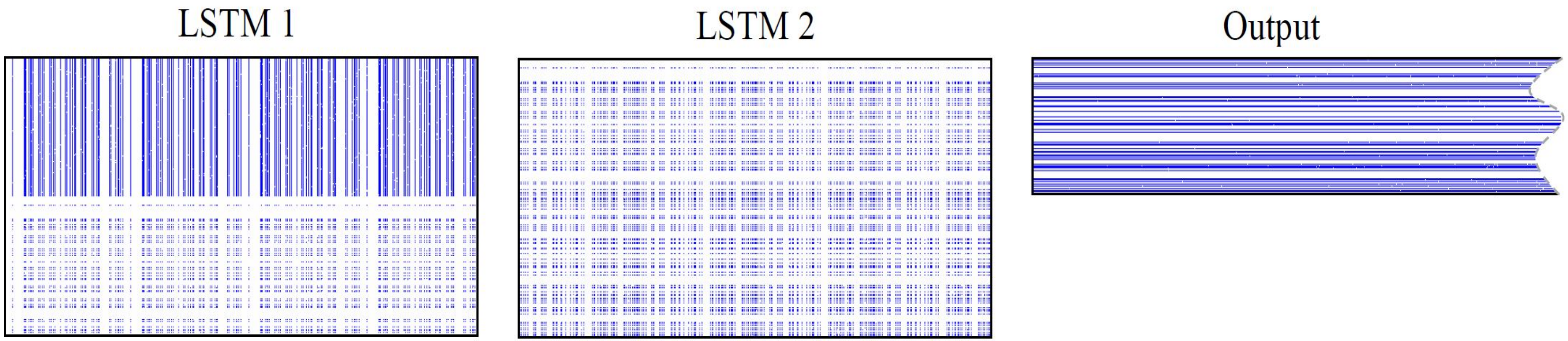
$$\mathbf{w}_k^{(n)} \leftarrow \mathbf{w}_k^{(n)} - \eta \cdot \left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_k^{(n)}} + \lambda \cdot \frac{\mathbf{w}_k^{(n)}}{\left|\left|\mathbf{w}_k^{(n)}\right|\right|_2} \right)$$



Weights matrices in LSTM

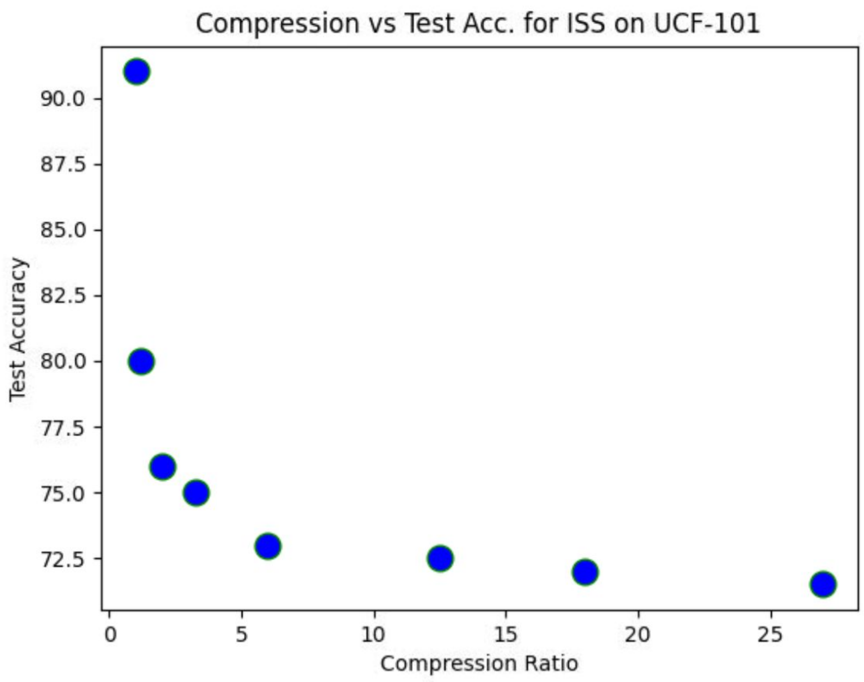
Intrinsic Sparse Structure (ISS) in LSTM

- Parameters Pruned through ISS compression is very much effective for multi-layer LSTM



ISS on UCF101 - ConvLSTM Compression Results

- The number of parameters reduce exponentially with trade-off in test accuracy.
- These results are without fine tuning of the pruned model. Fine tuning of the model increases test accuracy by 2% to 5% as compression ratio increases from 2 to 25.



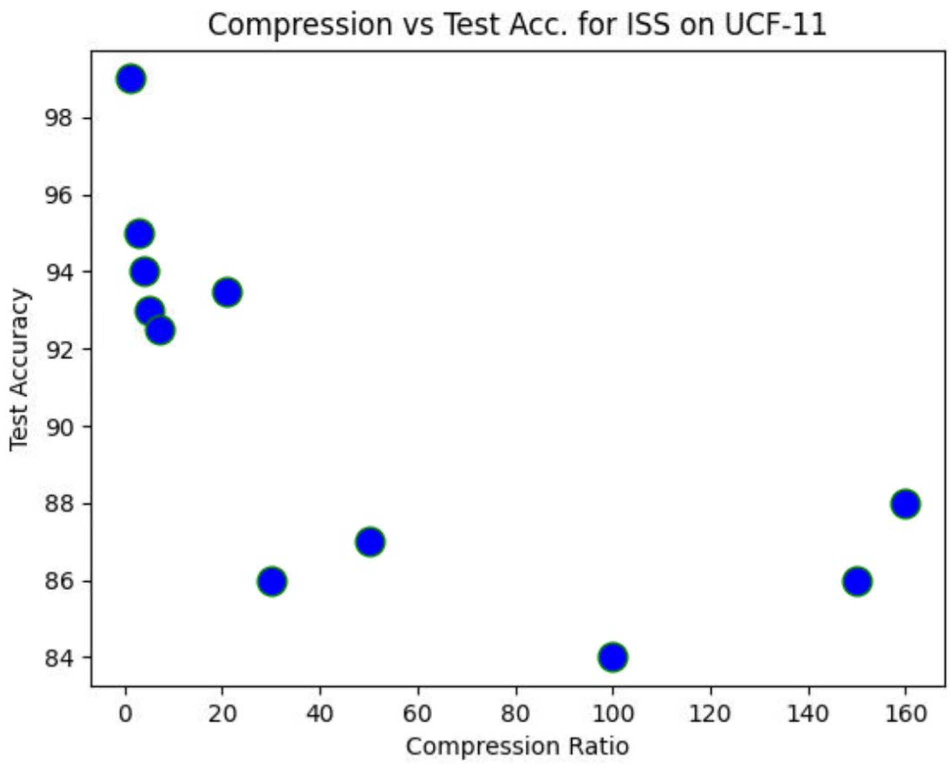
ISS on UCF101 - ConvLSTM Compression Results

- Uncompressed model has :
    - LSTM Compression ratio = 1
    - Latent dim = 512
    - Hidden dim = 1024
    - LSTM Params = 6.3 M
    - LSTM Size = 24.7 Mb
    - Top 1 Test Acc = 90 %

- Compressed model has :
    - LSTM Compression ratio = 27.4
    - Latent dim = 512
    - Hidden dim = 96
    - LSTM Params = 0.23 M
    - LSTM Size = 0.9 Mb
    - Top 1 Test Acc = 65 % (without fine-tuning)
    - Top 1 Test Acc = 70 % (with fine-tuning)

ISS on UCF11 - ConvLSTM Compression Results

- Similar behaviour as UCF101 except for fluctuations in acc and much larger compression ratio in this case.
- These results are without fine tuning of the pruned model. Fine tuning of the model increases test accuracy by 3% to 6% as compression ratio increases from 10 to 150.



ISS on UCF11 - ConvLSTM Compression Results

- Uncompressed model has :
  - LSTM Compression ratio = 1
  - Latent dim = 512
  - Hidden dim = 1024
  - LSTM Params = 6.3 M
  - LSTM Size = 24.7 Mb
  - Top 1 Test Acc = 99 %

- Compressed model has :
  - LSTM Compression ratio = 158
  - Latent dim = 512
  - Hidden dim = 17
  - LSTM Params = 0.04 M
  - LSTM Size = 0.157 Mb
  - Top 1 Test Acc = 89 % (without fine-tuning)
  - Top 1 Test Acc = 94% (with fine-tuning)

End-to-end-LSTM

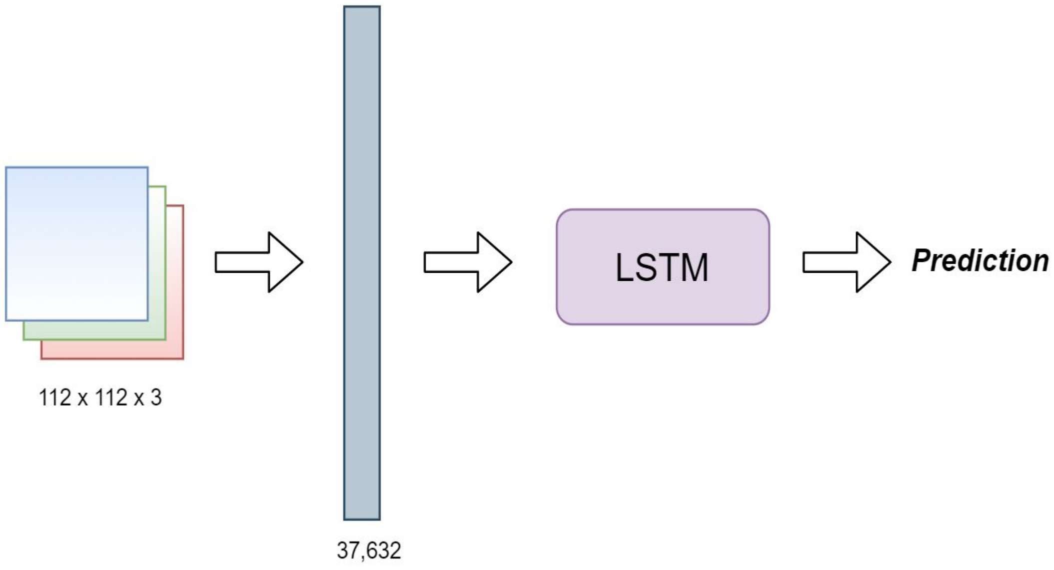
- Uncompressed E2E-LSTM Model

Latent Dim = 37632

Hidden Dim = 1024

LSTM Params = 158 M

Top 1 test Accuracy - 84.23 %



Further Work

- Need to automate or add in time frames constraint in objective function- such that inference can be done with minimum number of time frames - thus reducing inference time/flops.
- Validate compression theory on other sequential task such as speech recognition.
- Implementation of ISS with LSTM-VIB to get better compression numbers.
- Compression of end-to-end LSTMs of different sizes and comparison with other such benchmarks.

Conclusion

- Our method achieves large parameter reduction of LSTMs. In some cases, reduction in parameters lead to better accuracy than original model.
- It improves inference time and reduces memory footprint desired by applications on the edge.
- Currently tested and benchmarked on action recognition datasets.

References

1. Wen, Wei, et al. "Learning intrinsic sparse structures within long short-term memory." *arXiv preprint arXiv:1709.05027* (2017).
2. Dai, Bin, Chen Zhu, and David Wipf. "Compressing neural networks using the variational information bottleneck." *arXiv preprint arXiv:1802.10399* (2018).
3. Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *arXiv preprint arXiv:1510.00149* (2015).
4. Yu Pan et al. "Compressing Recurrent Neural Networks with Tensor Ring for Action Recognition", AAAI (2019)
5. Tjandra, Andros, Sakriani Sakti, and Satoshi Nakamura. "Recurrent Neural Network Compression Based on Low-Rank Tensor Representation." *IEICE Transactions on Information and Systems* 103.2 (2020): 435-449.
6. Thakker, Urmish, et al. "Pushing the limits of RNN Compression." *arXiv preprint arXiv:1910.02558* (2019).
7. Grachev, Artem M., Dmitry I. Ignatov, and Andrey V. Savchenko. "Compression of recurrent neural networks for efficient language modeling." *Applied Soft Computing* 79 (2019): 354-362.
8. Thakker, Urmish, et al. "Run-time efficient RNN compression for inference on edge devices." *2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. IEEE, 2019.
9. Winata, Genta Indra, et al. "On the effectiveness of low-rank matrix factorization for lstm model compression." *arXiv preprint arXiv:1908.09982* (2019).

Thank You

ISS +LSTM-VIB on UCF11 results