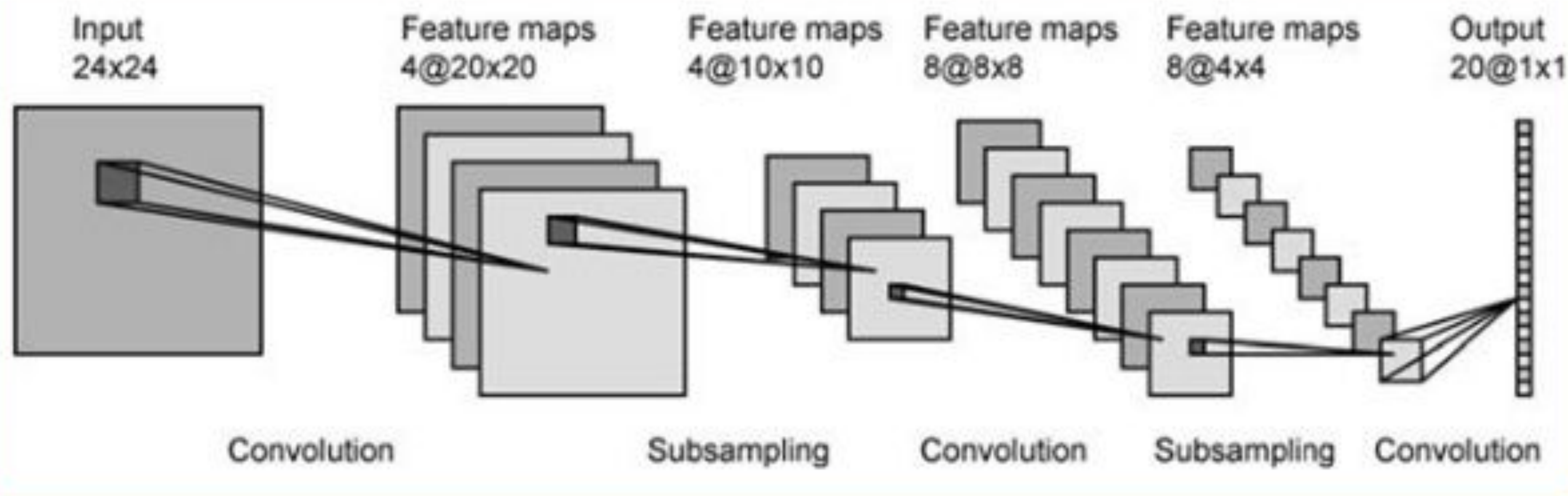
# XNOR-Nets: Binarized Neural Networks

*Sai Krishna Aditya Parvatha*
*Piyush Bhatt*
*Sameer Kumar Behera*
*Tanmay Verma*

# Convolutional Neural Network



# Convolution Operations

TEXAS A&M UNIVERSITY

| Model | LeNet-5 | AlexNet | VGG-16 | GoogleNet v1 | ResNet-5 |
|---|---|---|---|---|---|
| **Weights** | 60K | 61M | 138M | 7M | 25.5M |
| **MAC** | 341K | 724M | 15.5G | 1.43G | 3.9G |

**Source:** V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE. 105(12):2295–2329, Dec 2017

# INTRODUCTION
# TO
# BINARIZED NEURAL NETWORKS

# Binarized Convolutional Neural Network

- Weights are constrained to -1 (represented by bit value 0) or +1 (represented by bit value 1)
- Pre-activations are also constrained to -1 and 1
- Different popular attempts: Bitwise Neural Network, BinConnect, BinaryNet and XNOR-Net

# Matrix Multiplication vs XNOR

$$\begin{bmatrix}-1 & +1 & +1\\ -1 & -1 & -1\\ +1 & +1 & -1\end{bmatrix}\begin{bmatrix}-1\\ +1\\ +1\end{bmatrix} = \begin{bmatrix}(-1\mathrm{x}1) + (1\mathrm{x}1) + (1\mathrm{x}1)\\ (-1\mathrm{x}{-1}) + (1\mathrm{x}{-1}) + (1\mathrm{x}{-1})\\ (-1\mathrm{x}1) + (1\mathrm{x}1) + (1\mathrm{x}{-1})\end{bmatrix} = \begin{bmatrix}3\\ -1\\ -1\end{bmatrix}$$

$$\begin{bmatrix}0 & +1 & +1\\ 0 & 0 & 0\\ +1 & +1 & 0\end{bmatrix}\begin{bmatrix}0\\ +1\\ +1\end{bmatrix} = \begin{bmatrix}\mathrm{popcnt}(\mathrm{xnor}(011,011))\\ \mathrm{popcnt}(\mathrm{xnor}(011,000))\\ \mathrm{popcnt}(\mathrm{xnor}(011,110))\end{bmatrix} = \begin{bmatrix}3\\ -1\\ -1\end{bmatrix}$$

popcnt(xnor(011, 110)) = popcnt(xnor(0,1), xnor(1,1), xnor(1,0)) = popcnt(010) = -1 +1 -1 = -1

**Source: https://software.intel.com/en-us/articles/binary-neural-networks**

# BNN Advantages

- Small-sized model (32-bit weights replaced with a single-bit)
- Fast inference
- Power-efficient operations

# XNOR-Net

# XNOR-Net

- First Binary Neural Network that scales up to **ImageNet**

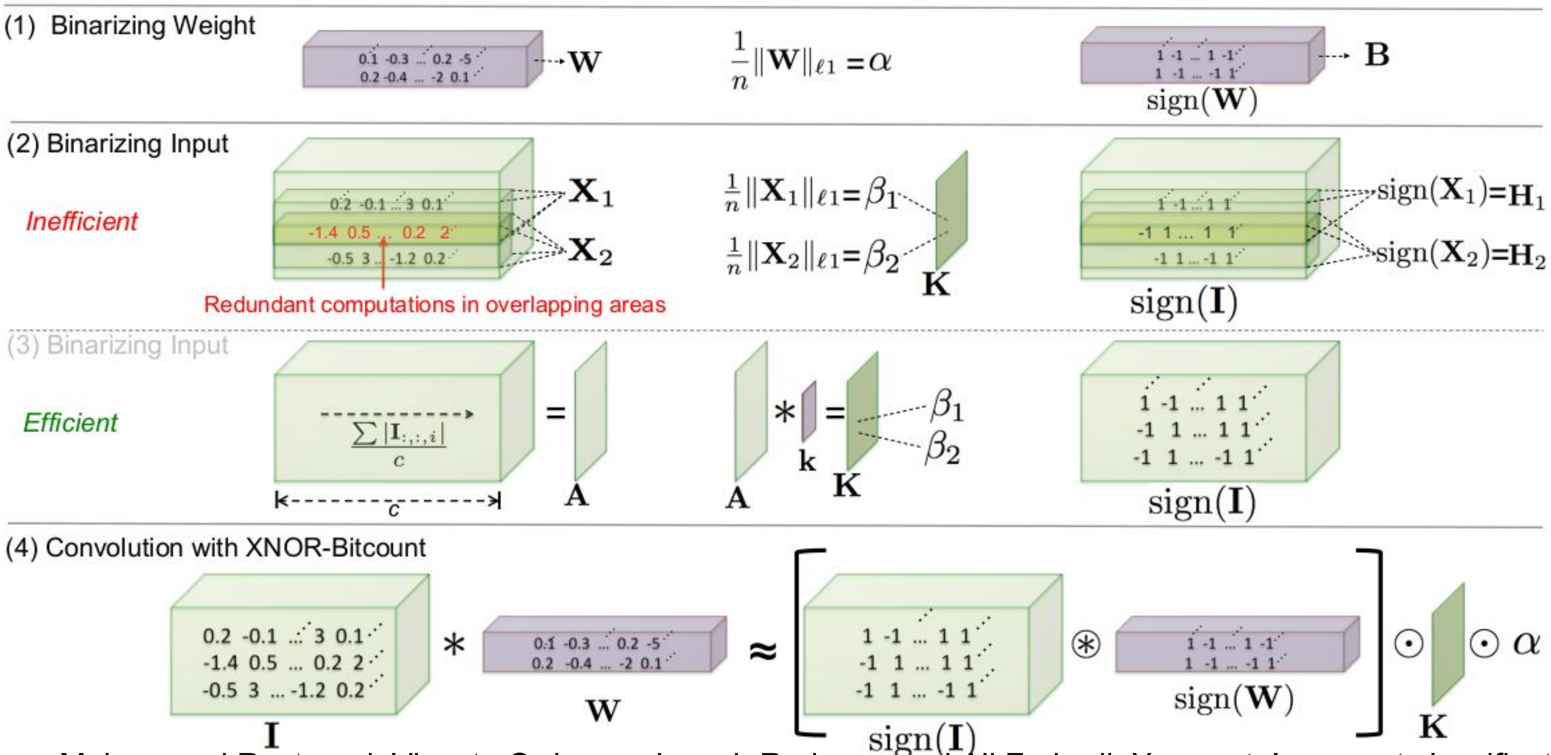
- A simple and efficient way of training and running inference on Binary Convolution Layers.

$$\alpha^*, \mathbf{B}^*, \beta^*, \mathbf{H}* = \underset{\alpha, \mathbf{B}, \beta, \mathbf{H}}{\operatorname{argmin}} \|\mathbf{X}^\mathsf{T}\mathbf{W} - \beta\alpha\mathbf{H}^\mathsf{T}\mathbf{B}\|$$

- **BOTTLENECK:** However, real-valued weights are still stored to calculate gradients and updating the weights.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. CoRR , abs/1603.05279, 2016.

# Convolution with XNOR-Net



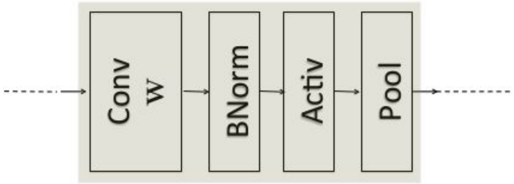

**Source:** Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. CoRR , abs/1603.05279, 2016.

# Important Guidelines to Preserve Accuracy

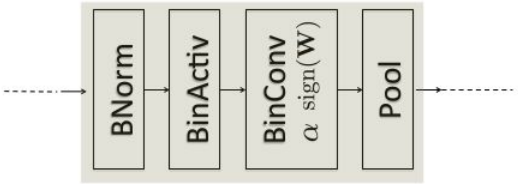


- The first convolution layer and last layer in the network are not binarized.

- The sequence of operations is slightly modified from traditional CNN block




A typical block in CNN




A block in XNOR-Net

# Our Project Includes

TEXAS A&M UNIVERSITY

- Implementation of XNOR-Nets to compare accuracy loss against the full-precision Neural Networks
- Filter and Activation Visualization to see whether Binarization actually works and gain better insights and validate our implementation
- Evaluate the discussed Binarization Technique on Textual Dataset (Sentiment Analysis)

# CLASSIFICATION PROBLEM
# IN
# A SMALL IMAGE DATASET
# (MNIST)

# PROBLEM STATEMENT

TEXAS A&M UNIVERSITY

- Hand-written Digit Recognition on MNIST Dataset
- MNIST consists of:
  - Training Set of 60000 examples
  - Test Set of 10000 examples
- Considered as ‘Hello World’ of Machine Learning
- Will visualize the activations and filters to validate our implementation

# Our LeNet-5 Model

TEXAS A&M UNIVERSITY

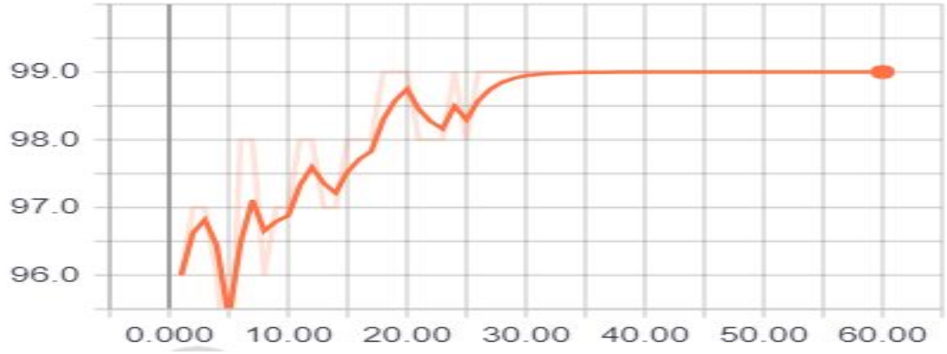
```
LeNet_5(
  (conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
  (bn_conv1): BatchNorm2d(20, eps=0.0001, momentum=0.1, affine=False, track_running_stats=True)
  (relu_conv1): ReLU(inplace)
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (bin_conv2): BinConv2d(
        (bn): BatchNorm2d(20, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
        (conv): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
        (k_conv): Conv2d(1, 1, kernel_size=(5, 5), stride=(1, 1)) <= Fixed-filter convolution to obtain the Beta values
        (relu): ReLU(inplace)
  )
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (bin_ip1): BinConv2d(
        (bn): BatchNorm2d(50, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
        (linear): Linear(in_features=800, out_features=500, bias=True)
        (relu): ReLU(inplace)
  )
  (ip2): Linear(in_features=500, out_features=10, bias=True)
```

**Source:** http://yann.lecun.com/exdb/lenet/

# Validation Accuracy and Loss



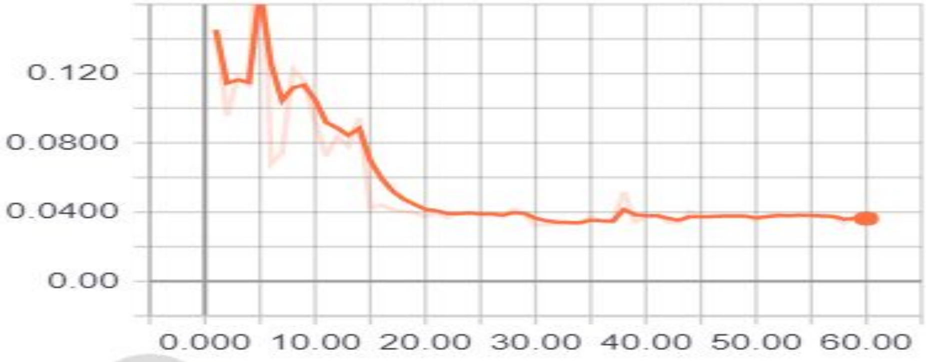
Accuracy




Max: 99.24%

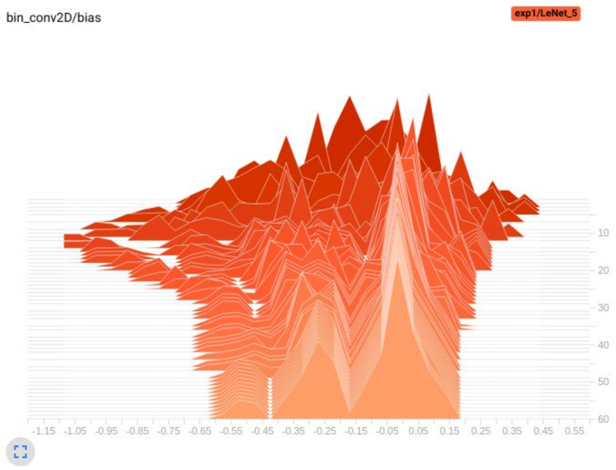
Epochs ->

Loss



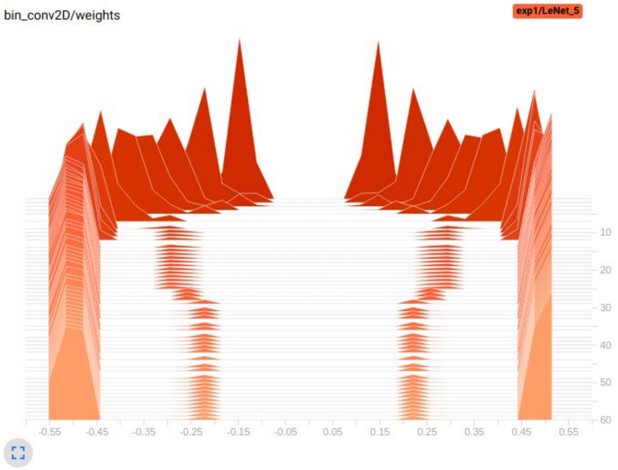

Min: 0.036

# Bias and Weights Histograms



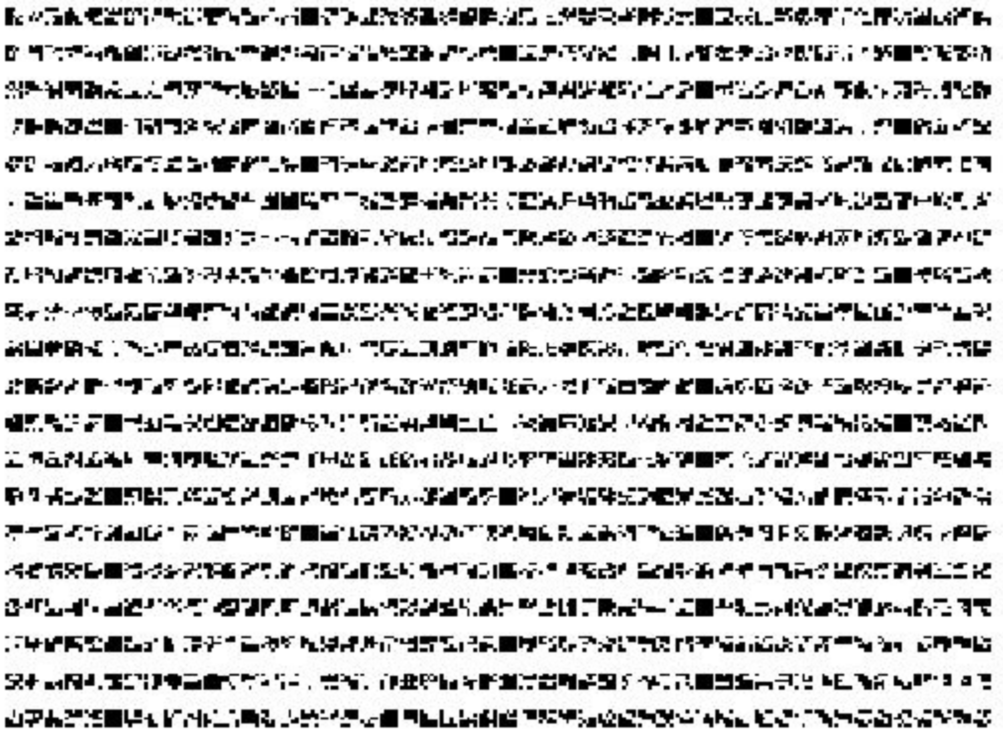

Bias

Epochs




Weights

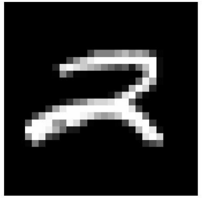
# Learned Kernels in bin_conv2



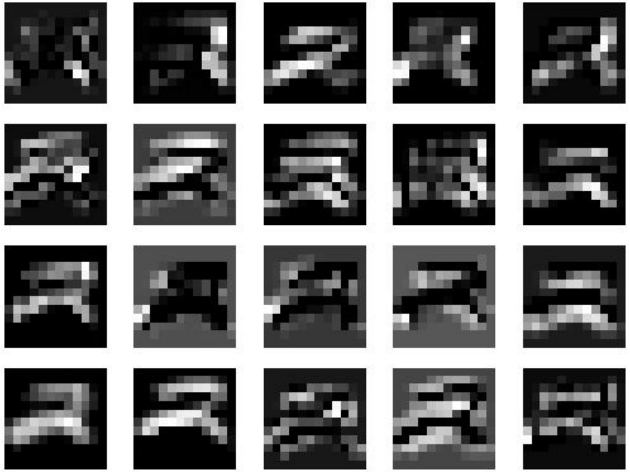
bin_conv2 have 20 input channels and 50 output channels. Hence, there are 1000 kernels.

**NOTE:** All the weights are binary, either white or black.

# Output of conv1 (Full-precision)



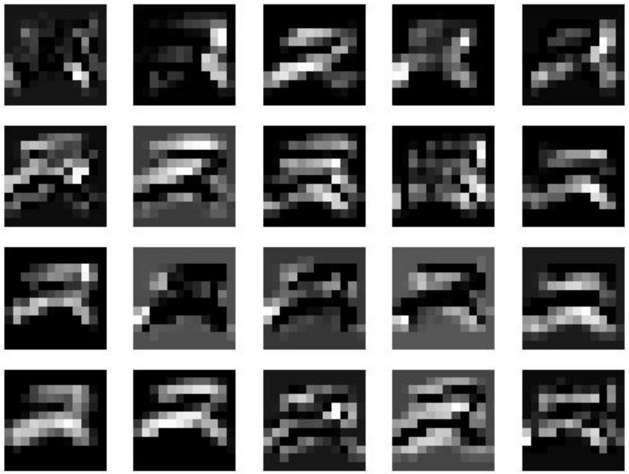
Input



Output of conv1 (full-precision convolution)

# Binary Activation



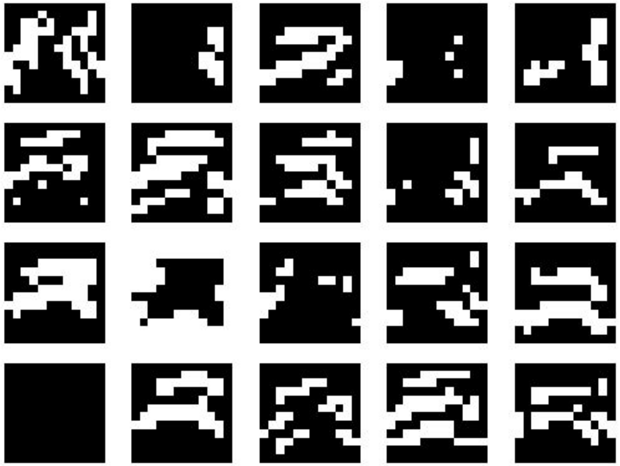


Output of first convolution block

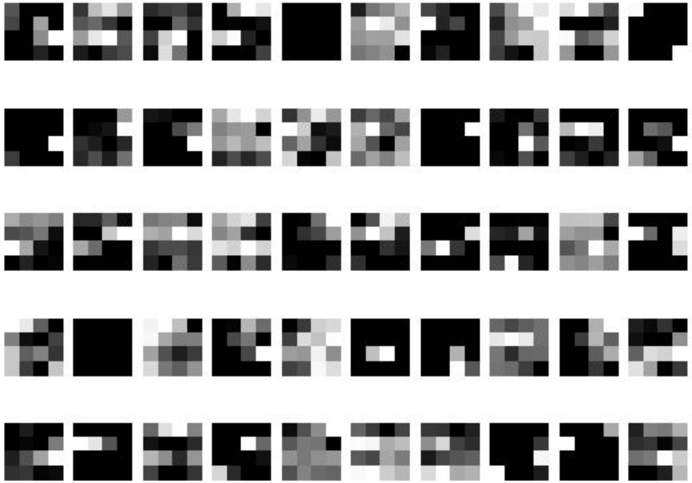
Input to bin_conv2 which is either 0 or 1

**NOTE:** At this point we have verified that both the inputs and the kernel weights to bin_conv2 are either 0 or 1

# Output of bin_conv2



Input to bin_conv2 which is either 0 or 1



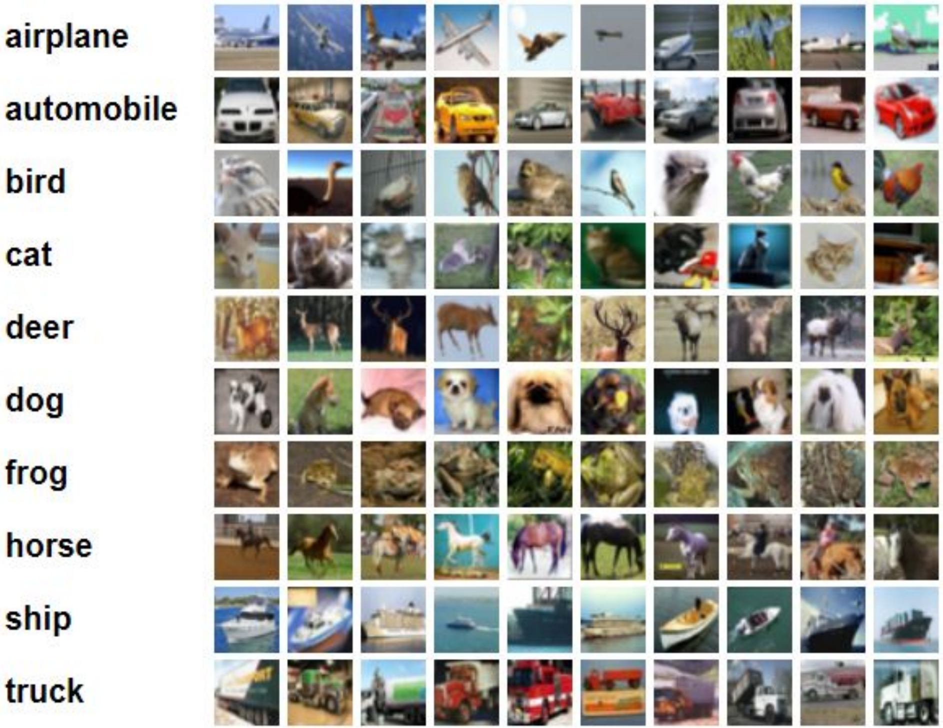
The output of bin_conv2

# CLASSIFICATION PROBLEM IN A LARGE IMAGE DATASET (CIFAR-10)

# PROBLEM STATEMENT

TEXAS A&M UNIVERSITY

- Image Classification on CIFAR-10
- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.
- There are 50000 training images and 10000 test images.
- The dataset is divided into 5 training batches and 1 test batch.
- The classes are completely mutually exclusive.



# Network-In-Network (NIN) Model

```
(module): Net(
    (xnor): Sequential(
    (0): Conv2d(3, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=False, track_running_stats=True)
    (2): ReLU(inplace)
    (3): BinConv2d(
    (bn): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
    (conv): Conv2d(192, 160, kernel_size=(1, 1), stride=(1, 1))
    (relu): ReLU(inplace)
    )
    (4): BinConv2d(
    (bn): BatchNorm2d(160, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
    (conv): Conv2d(160, 96, kernel_size=(1, 1), stride=(1, 1))
    (relu): ReLU(inplace)
    )
    (5): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
```

Source: Lin, M., Chen, Q., & Yan, S. (2013). “Network In Network”. CoRR, abs/1312.4400.

# Network-In-Network (NIN) Model

```
(6): BinConv2d(
(bn): BatchNorm2d(96, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
(dropout): Dropout(p=0.5)
(conv): Conv2d(96, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
(relu): ReLU(inplace)
)
(7): BinConv2d(
(bn): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
(conv): Conv2d(192, 192, kernel_size=(1, 1), stride=(1, 1))
(relu): ReLU(inplace)
)
(8): BinConv2d(
(bn): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
(conv): Conv2d(192, 192, kernel_size=(1, 1), stride=(1, 1))
(relu): ReLU(inplace)
)
(9): AvgPool2d(kernel_size=3, stride=2, padding=1)
```

Source: Lin, M., Chen, Q., & Yan, S. (2013). “Network In Network”. CoRR, abs/1312.4400.

# Network-In-Network (NIN) Model

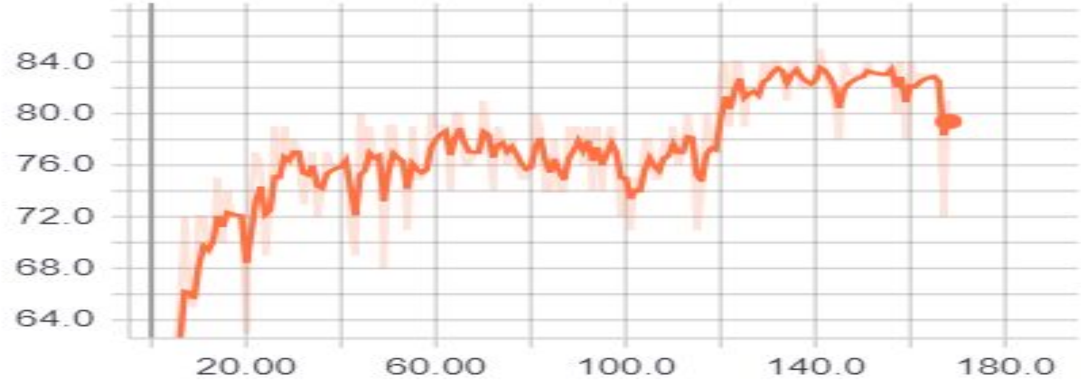
```
(10): BinConv2d(
(bn): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
(dropout): Dropout(p=0.5)
(conv): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(relu): ReLU(inplace)
)
(11): BinConv2d(
(bn): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=True, track_running_stats=True)
(conv): Conv2d(192, 192, kernel_size=(1, 1), stride=(1, 1))
(relu): ReLU(inplace)
)
(12): BatchNorm2d(192, eps=0.0001, momentum=0.1, affine=False, track_running_stats=True)
(13): Conv2d(192, 10, kernel_size=(1, 1), stride=(1, 1))
(14): ReLU(inplace)
(15): AvgPool2d(kernel_size=8, stride=1, padding=0)
)
)
```

Source: Lin, M., Chen, Q., & Yan, S. (2013). “Network In Network”. CoRR, abs/1312.4400.

# Validation Accuracy and Loss

TEXAS A&M UNIVERSITY

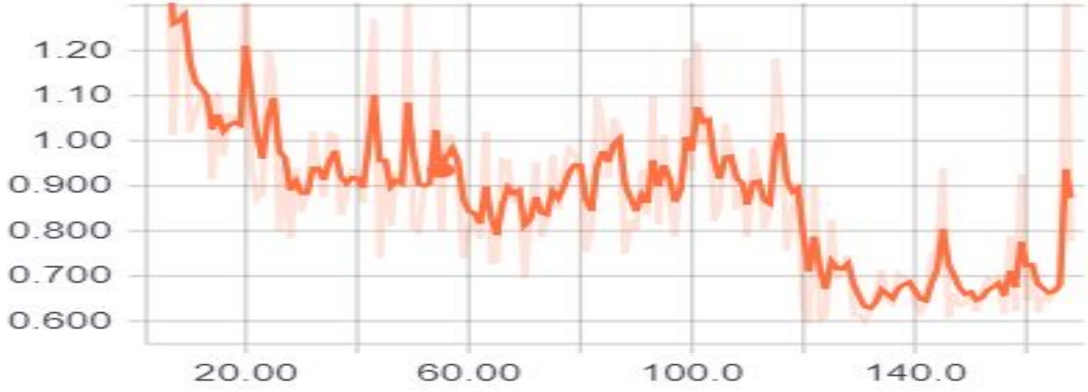
Accuracy




Epochs ->

**Max:** 85.02%

Loss

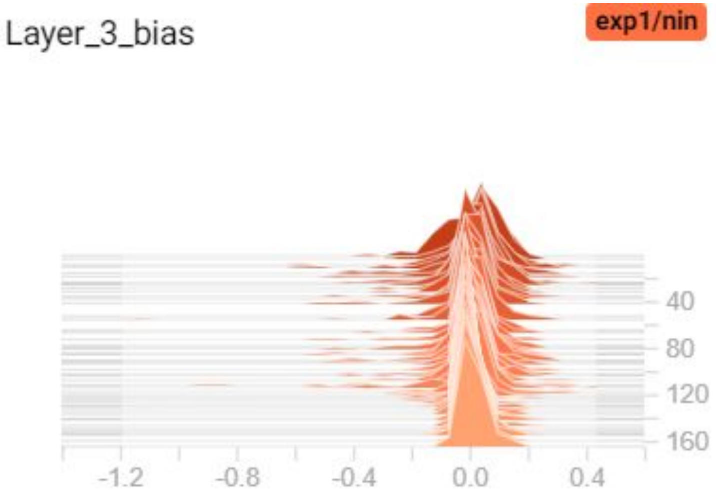



**Min:** 0.595

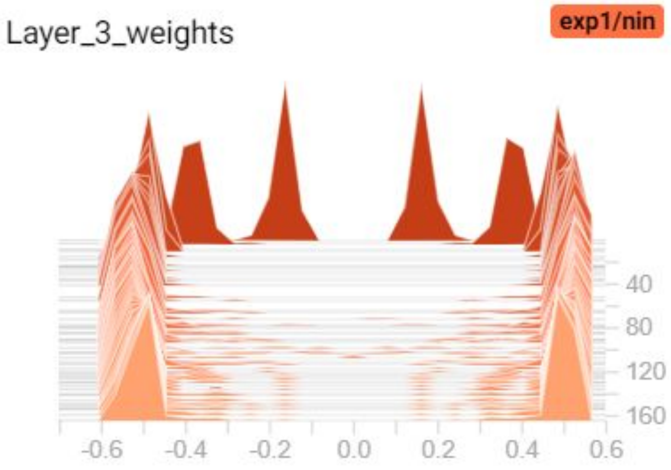
# Weights and Bias Histograms



## Layer 3 - 1st Binary Convolution




Epochs

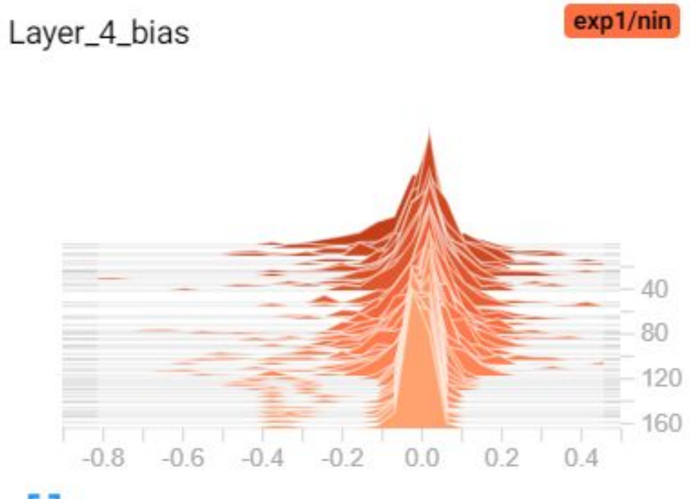



Bias

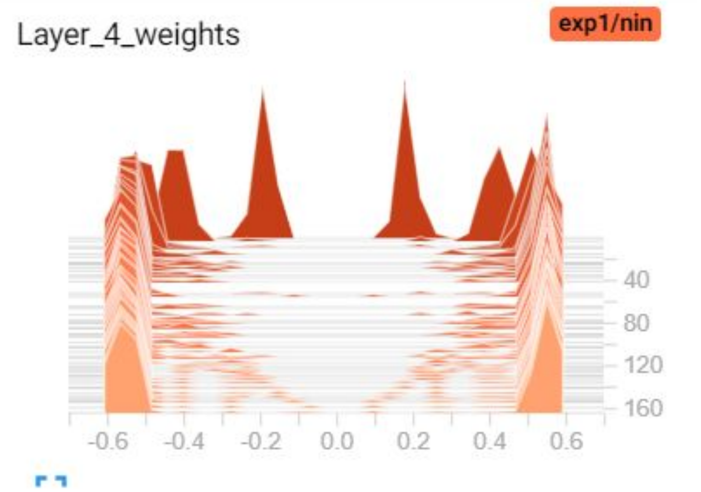
Weights

# Weights and Bias Histograms

## Layer 4 - 2nd Binary Convolution




Epochs



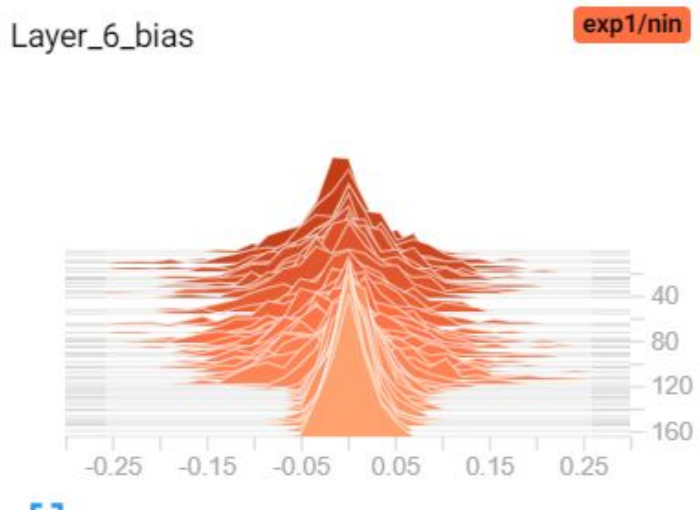

Bias

Weights

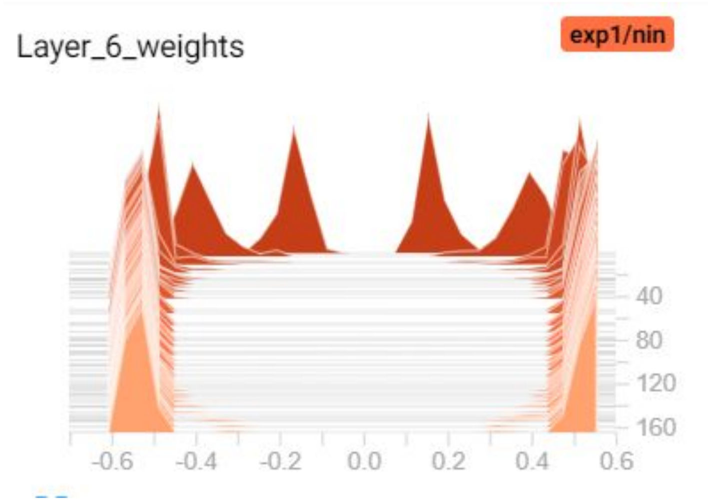
# Weights and Bias Histograms

TEXAS A&M UNIVERSITY

## Layer 6- 3rd Binary Convolution




Epochs

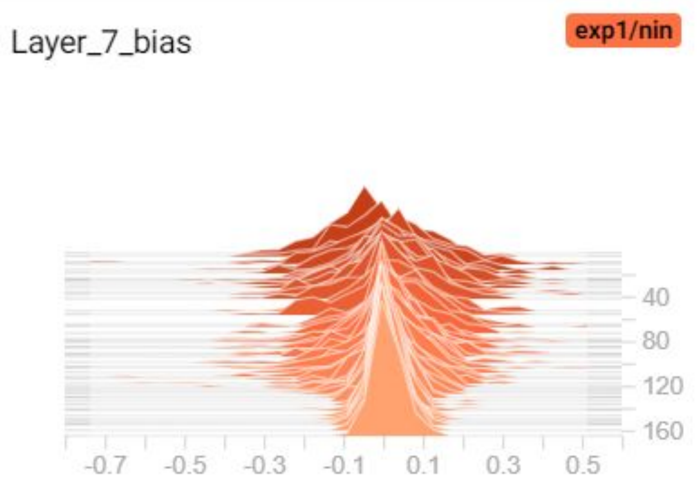



Bias

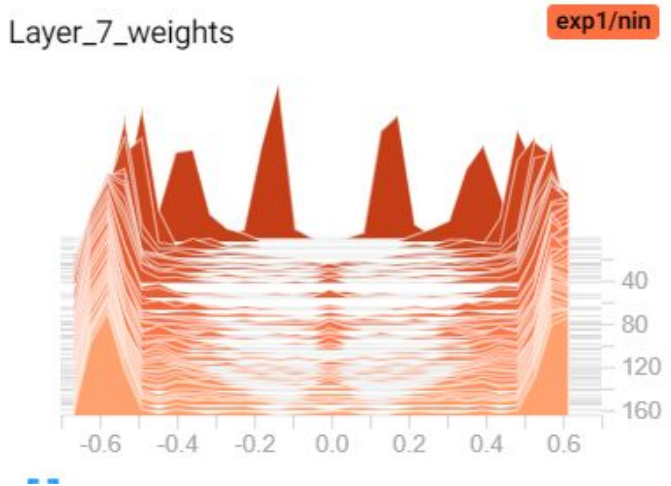
Weights

# Weights and Bias Histograms

## Layer 7 - 4th Binary Convolution




Epochs



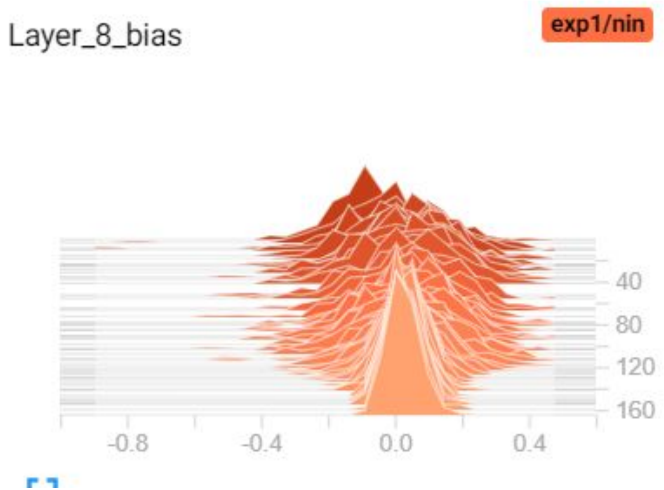

Bias

Weights

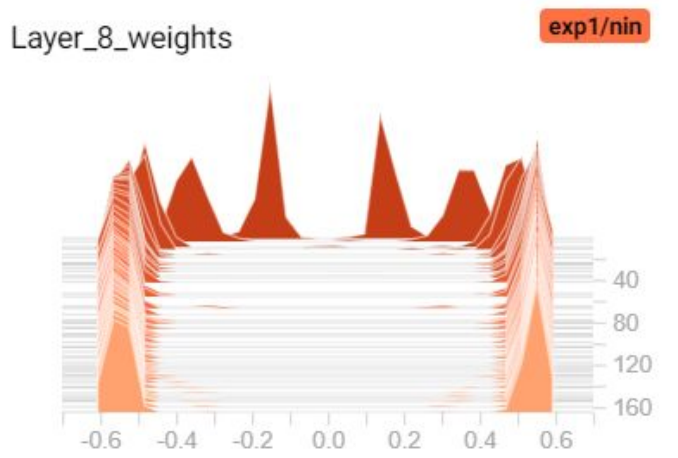
# Weights and Bias Histograms

TEXAS A&M UNIVERSITY

## Layer 8 - 5th Binary Convolution




Epochs



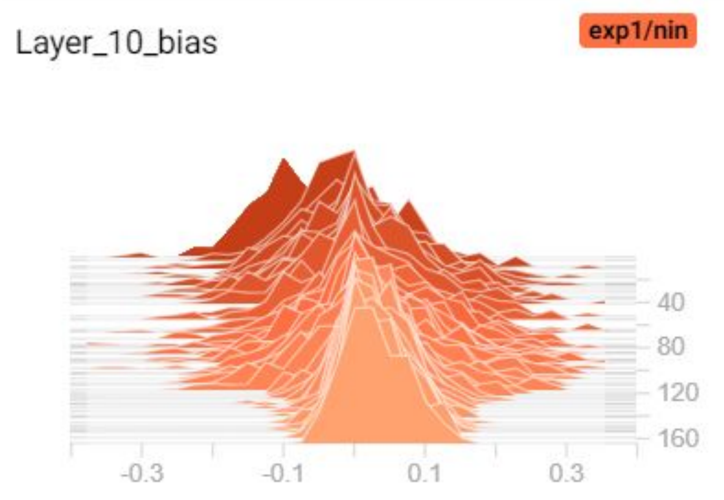

Bias

Weights

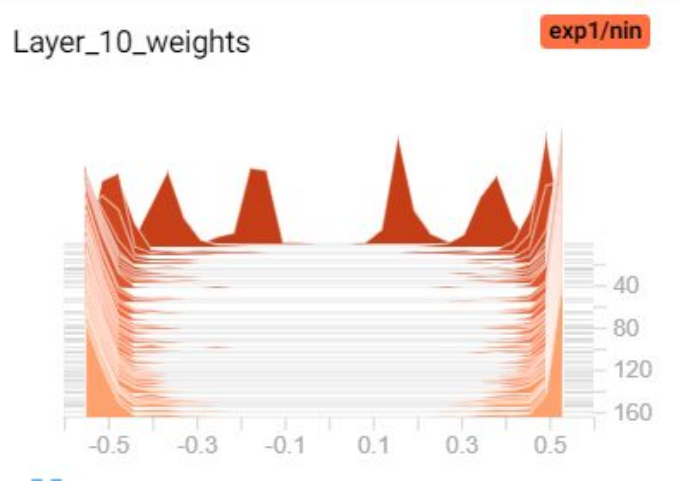
# Weights and Bias Histograms

TEXAS A&M UNIVERSITY

## Layer 10- 6th Binary Convolution




Epochs



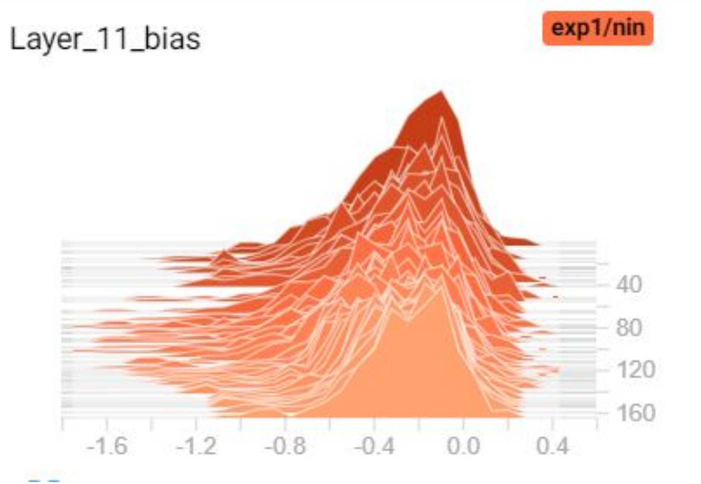

Bias

Weights

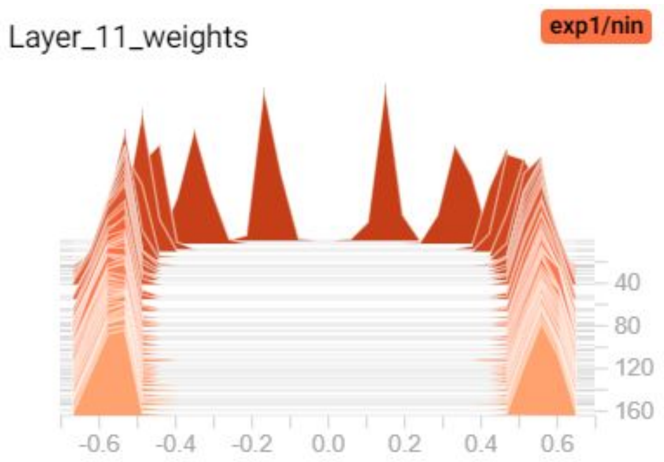
# Weights and Bias Histograms

TEXAS A&M UNIVERSITY

## Layer 11 - 7th Binary Convolution




Epochs




Bias

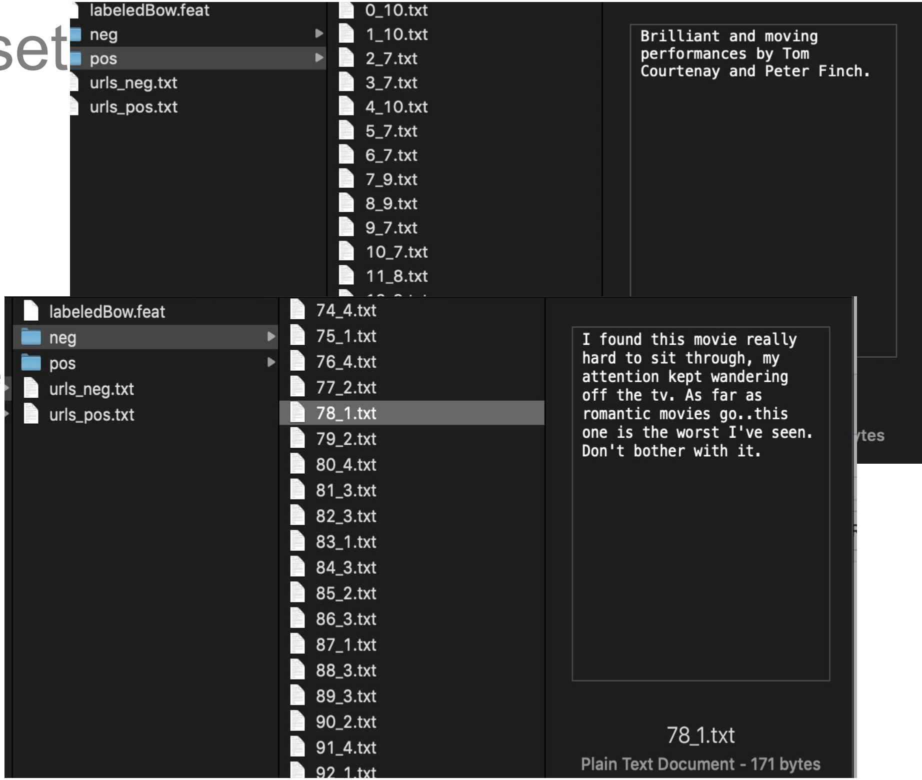
Weights

# CLASSIFICATION PROBLEM
# IN
# A TEXT DATASET
# (IMDB)

# Dataset

## IMDB Movie Review Text Dataset

- Movie Reviews
- Labels : Binary Sentiment
- 50k labelled reviews
- 50k unlabelled reviews
- At most 30 reviews for a single movie
    - Correlation of reviews
- Stratified train and test set
    - Disjoint on the basis of movies
- Sentiment Analysis

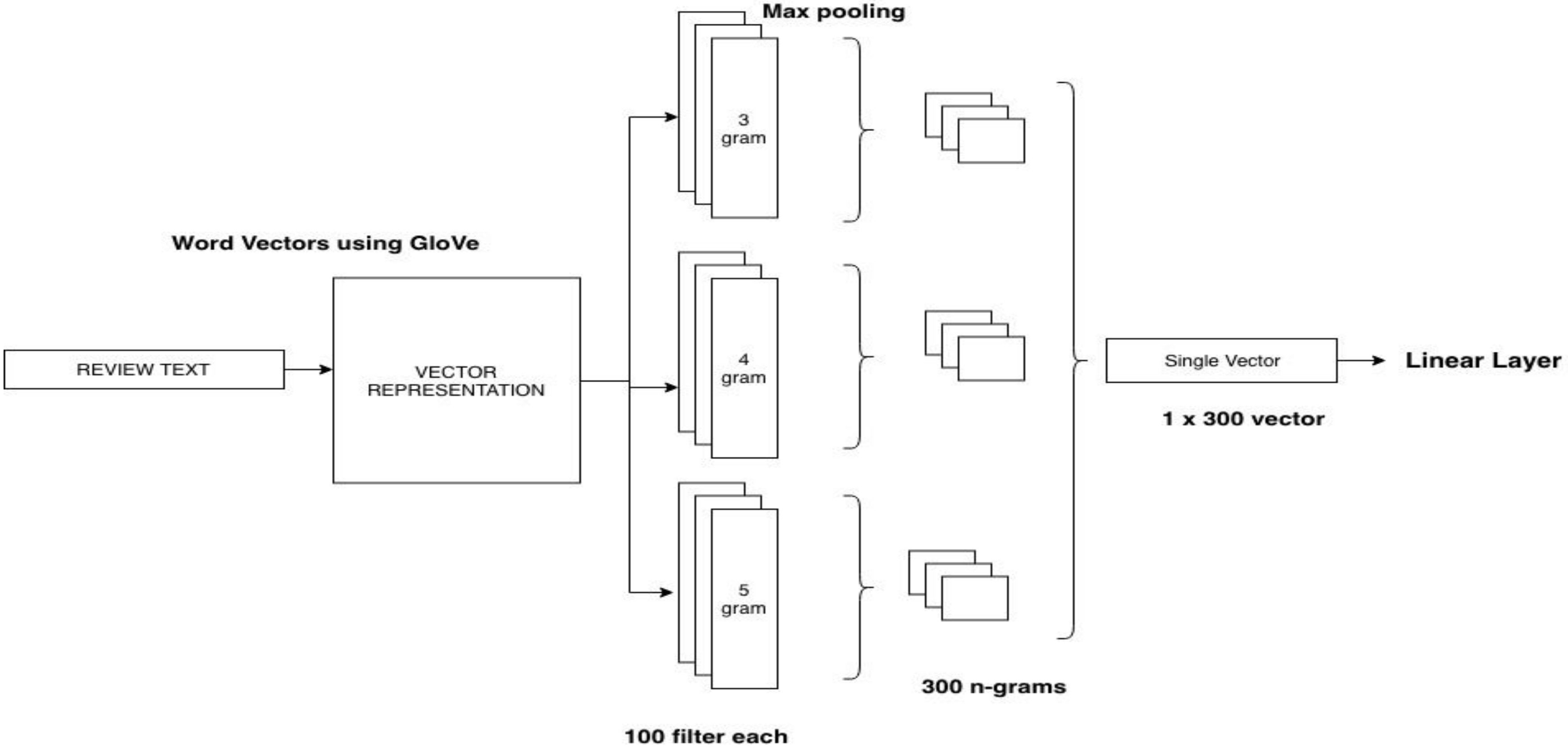


# GloVe

## GloVe - **Glo**bal **Ve**ctors for Word Representation

- Used to obtain vector representation for words
- Pre-trained word vector
  - **glove.6B**
    - 50d
    - **100d (400K vocab,6B tokens)**
    - 200d
    - 300d
  - glove.42B.300d
  - glove.840B.300d
  - gove.twitter.27B

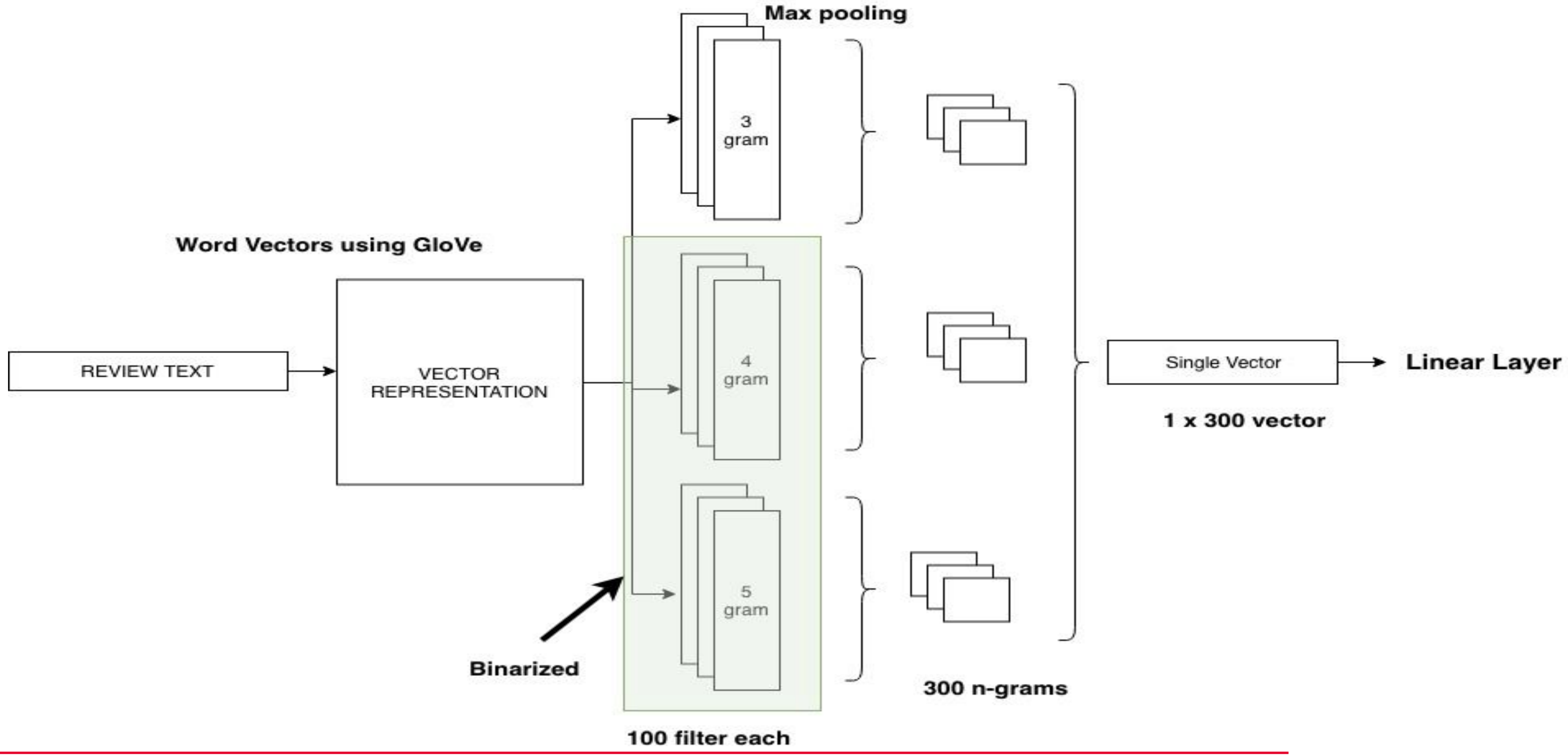
# Model




Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. EMNLP.

# Binarization

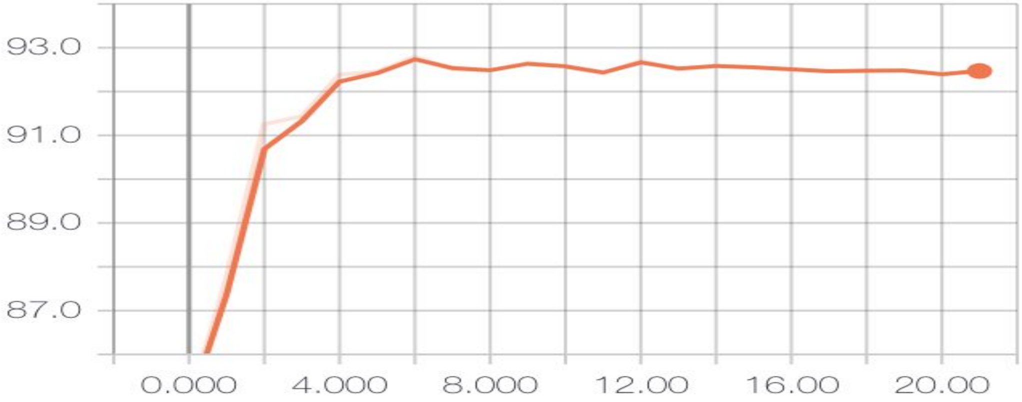




# Validation Accuracy and Loss

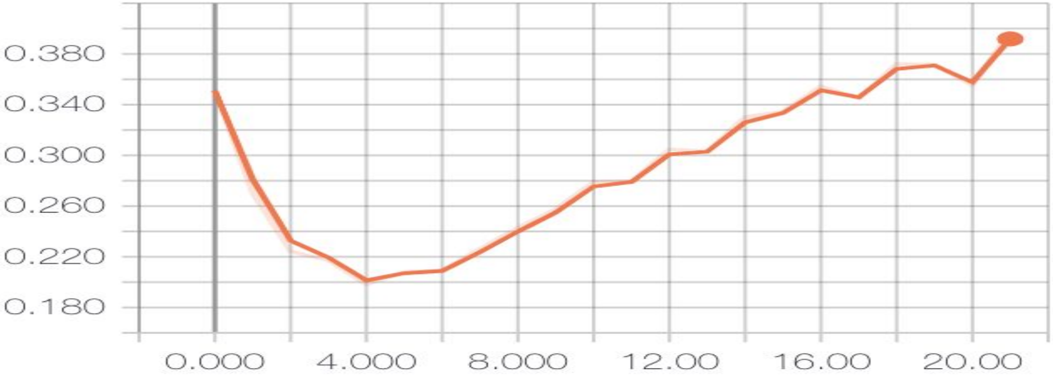


Accuracy



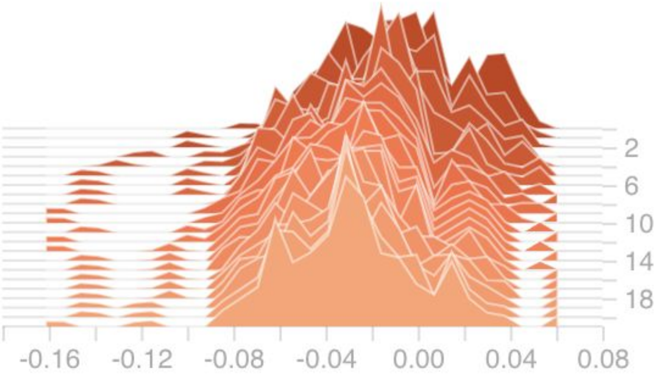

Epochs ->

Loss

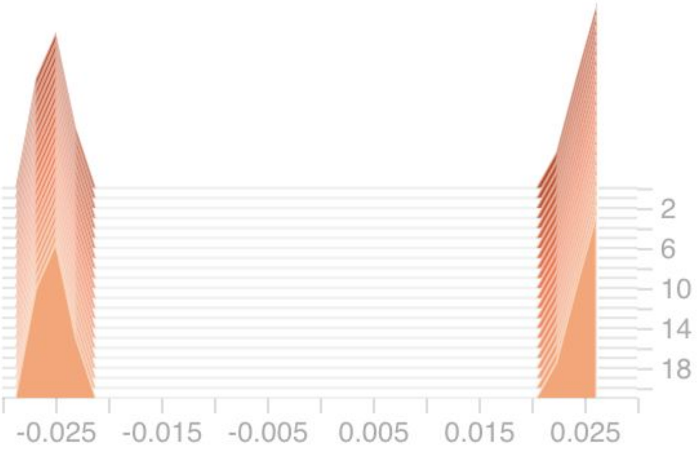


# Weights and Bias Histograms

## 4-gram




Epochs

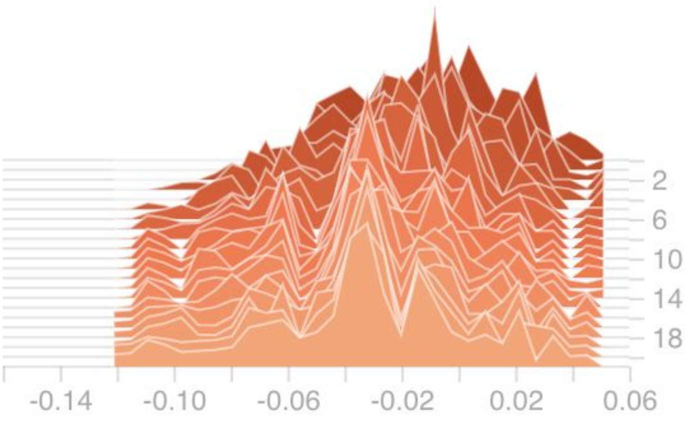



Bias

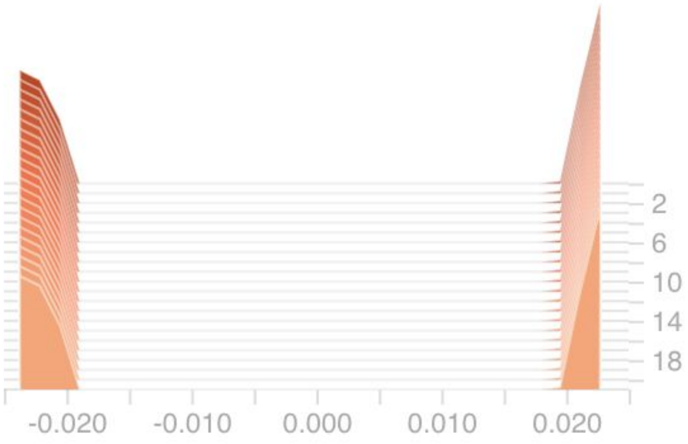
Weights

# Weights and Bias Histograms

**5-gram**




Epochs

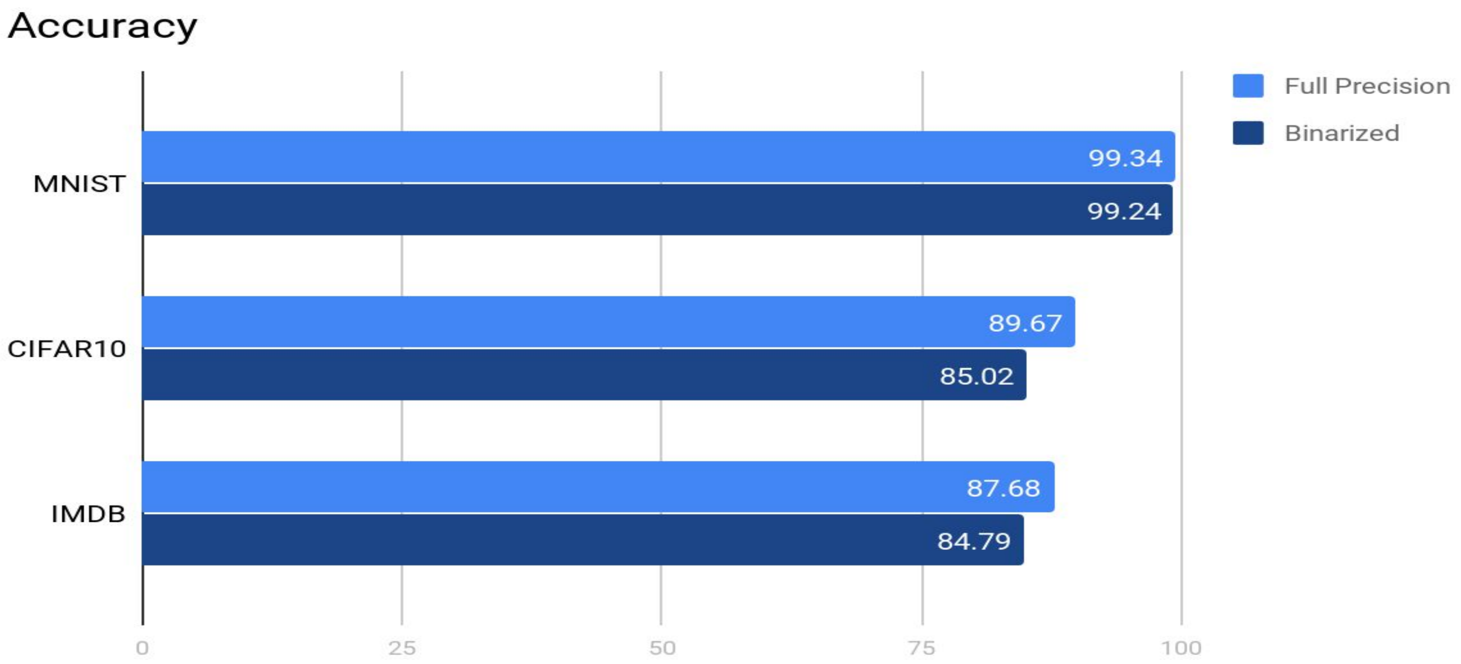



Bias

Weights

# Final Results





# Future Work

# Future Work

TEXAS A&M UNIVERSITY

- ImageNet Dataset
- I2B2 Dataset
- Optimizing with actual Bitwise Operations on GPU and how the binary values are saved.

# THANK YOU

## Questions?