Advanced NLP

Summer 2023

Anoop Sarkar

LayerNorm

https://arxiv.org/abs/1607.06450

also see: https://arxiv.org/abs/1911.07013

$$\mu = \frac{1}{H}\sum_{i=1}^{H} x_i \qquad \sigma^2 = \frac{1}{H}\sum_{i=1}^{H} (x_i - \mu)^2$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_H)$$

$$N(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sigma + \epsilon}$$

ϵ avoids div by zero

$$\mathbf{h} = \mathbf{g} \cdot N(\mathbf{x}) + \mathbf{b}$$

g and **b** are hyperparameters with dimension H

In PyTorch

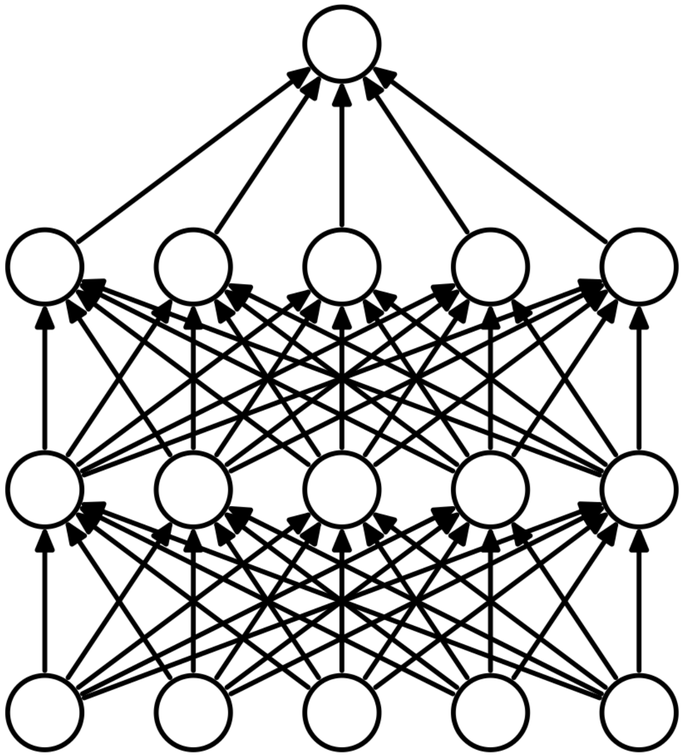
```
>>> # NLP Example
>>> batch, sentence_length, embedding_dim = 20, 5, 10
>>> embedding = torch.randn(batch, sentence_length, embedding_dim)
>>> layer_norm = nn.LayerNorm(embedding_dim)
>>> # Activate module
>>> layer_norm(embedding)
```

Dropout

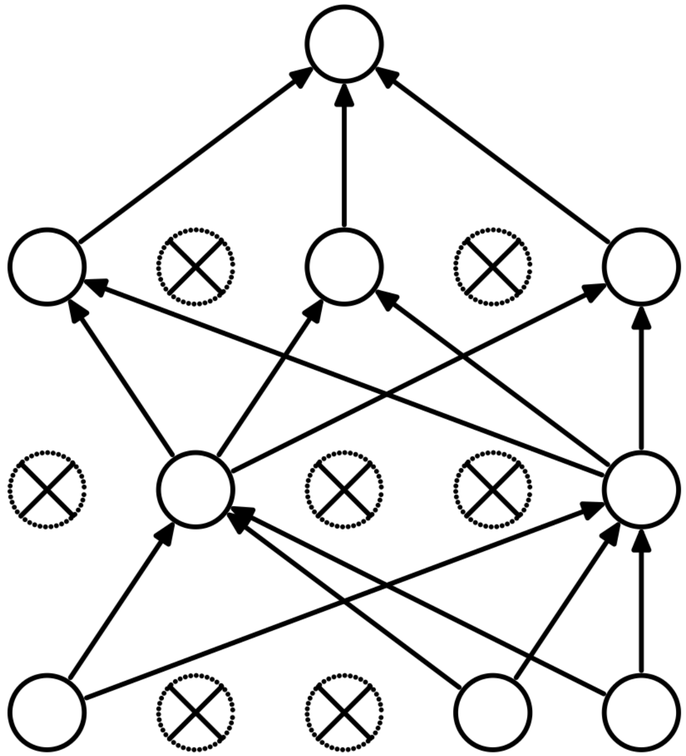
https://jmlr.org/papers/v15/srivastava14a.html

https://arxiv.org/abs/1207.0580

aka how to train 2^n neural networks when it has n units



(a) Standard Neural Net



(b) After applying dropout.

Before dropout

$$
\begin{aligned}
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}),
\end{aligned}
$$

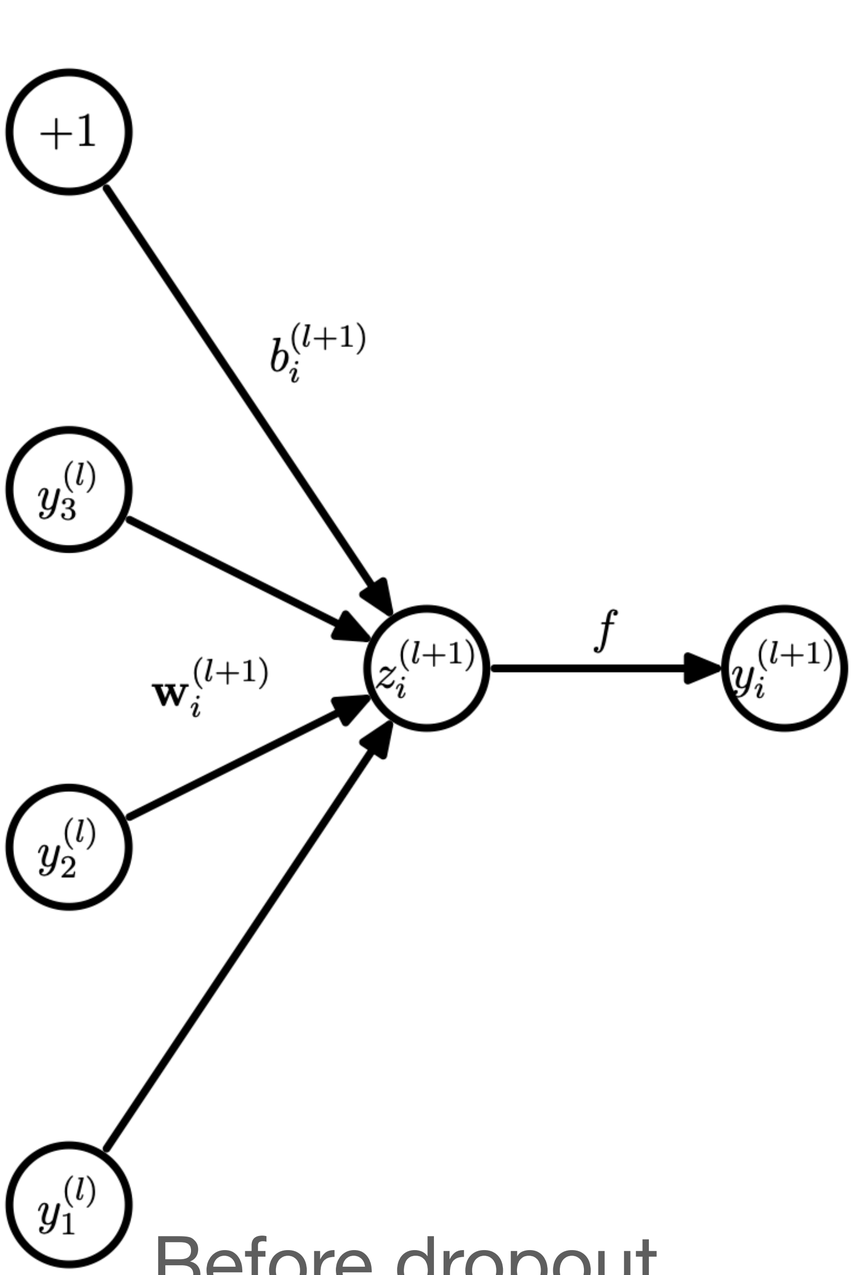
After dropout

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

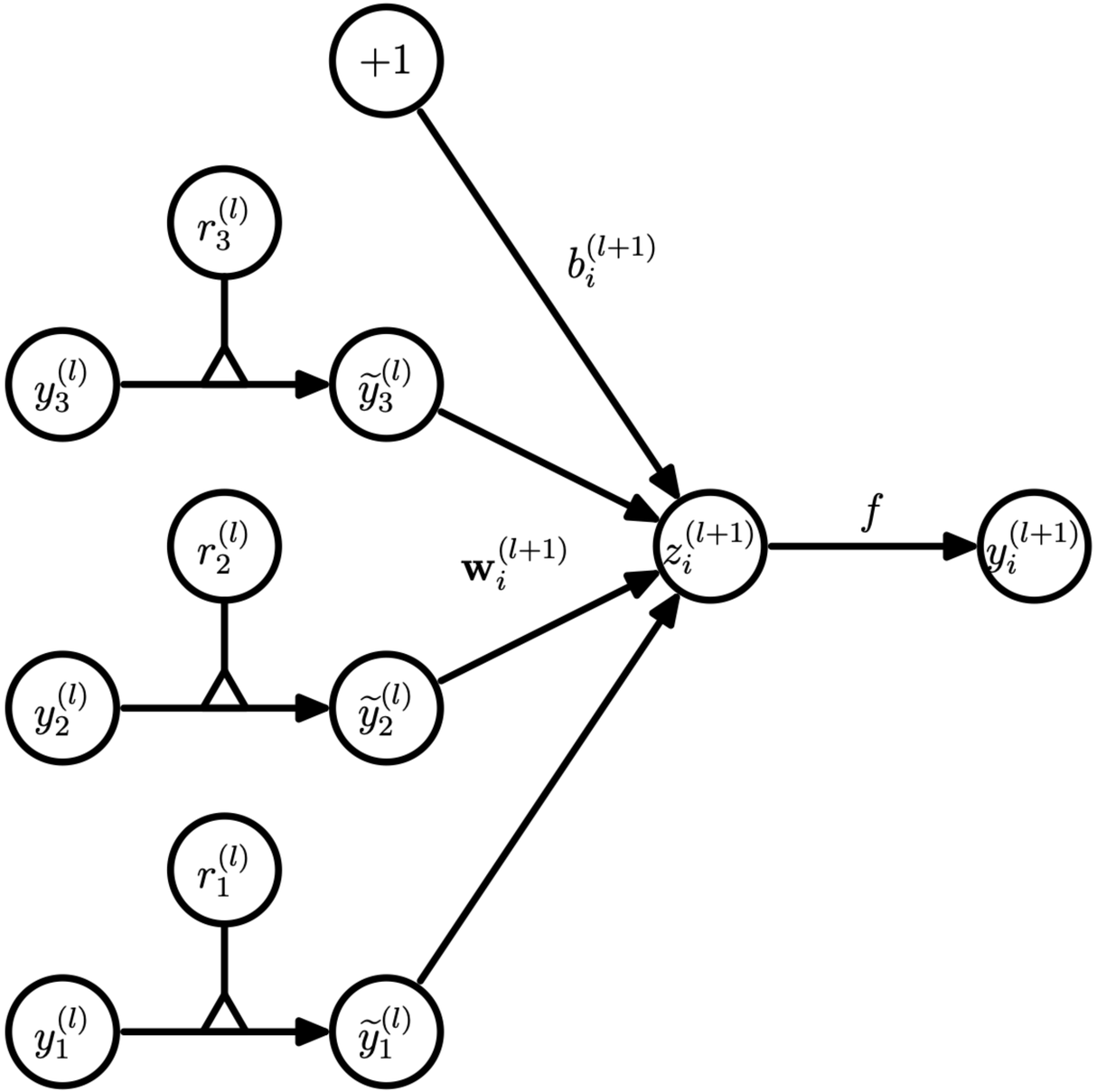
In PyTorch

```
>>> m = nn.Dropout(p=0.2)
>>> input = torch.randn(20, 16)
>>> output = m(input)
```

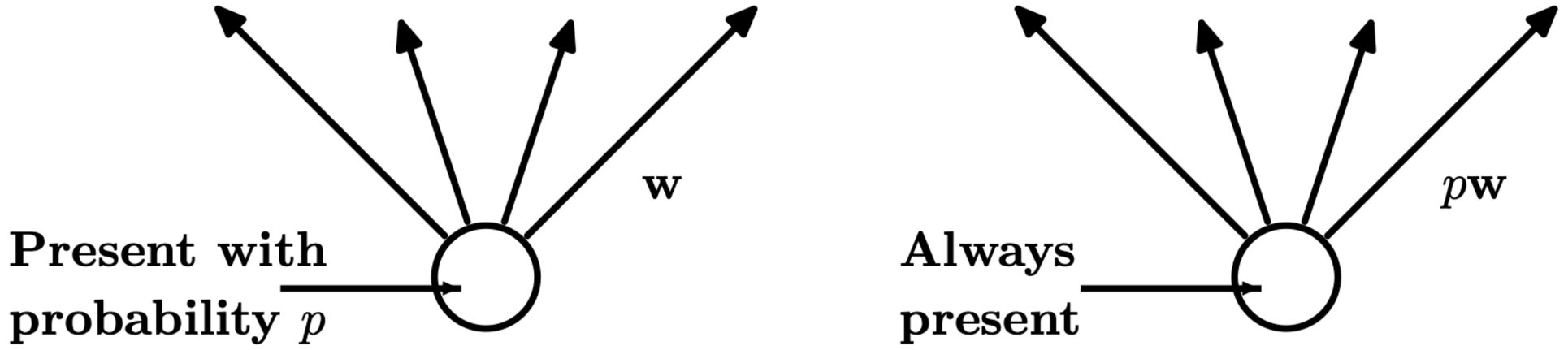
default: 0.5



Before dropout



After dropout



(a) At training time (b) At test time

Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

In Pytorch the outputs are scaled by a factor of $\frac{1}{1-p}$ during training so at inference/ test/evaluation time the dropout function simply computes the identity function

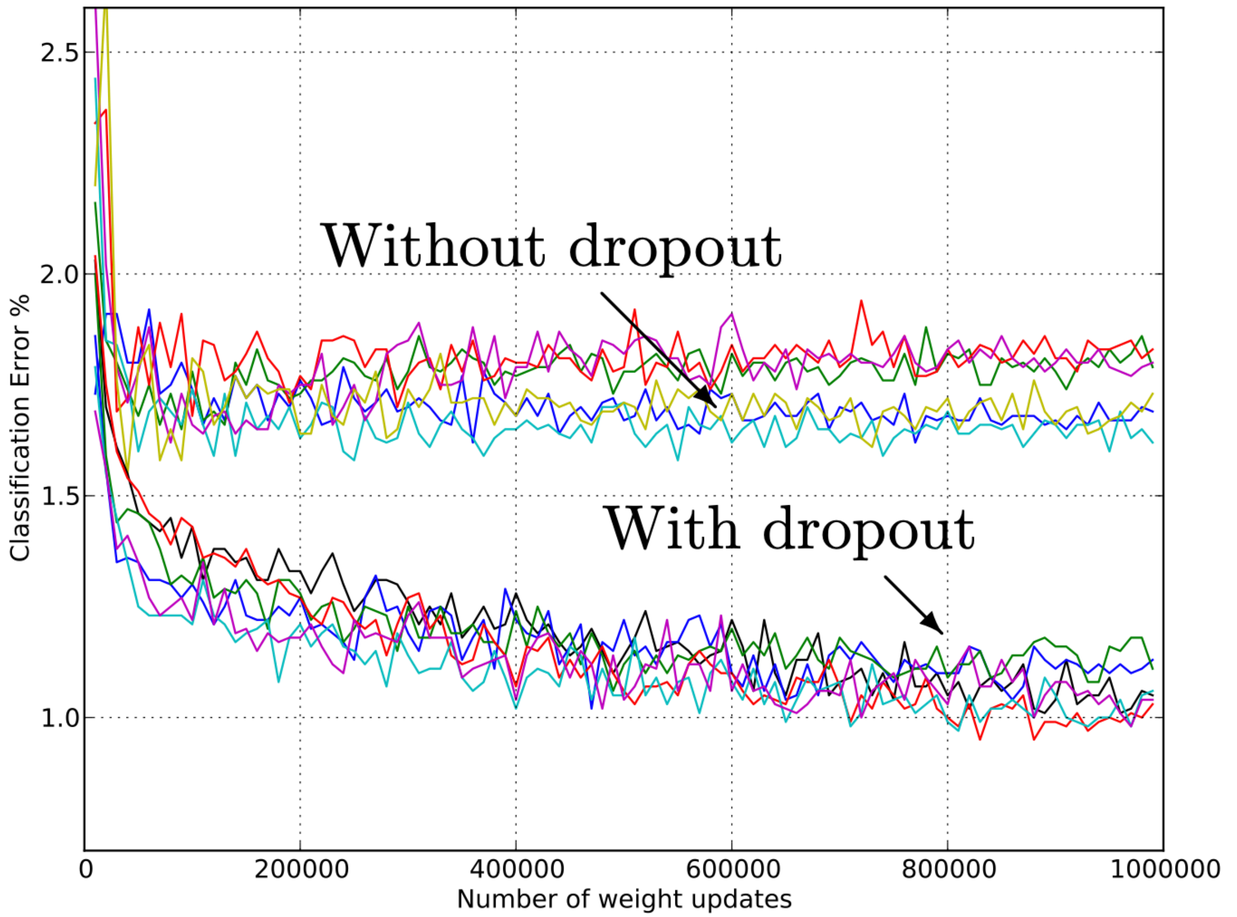
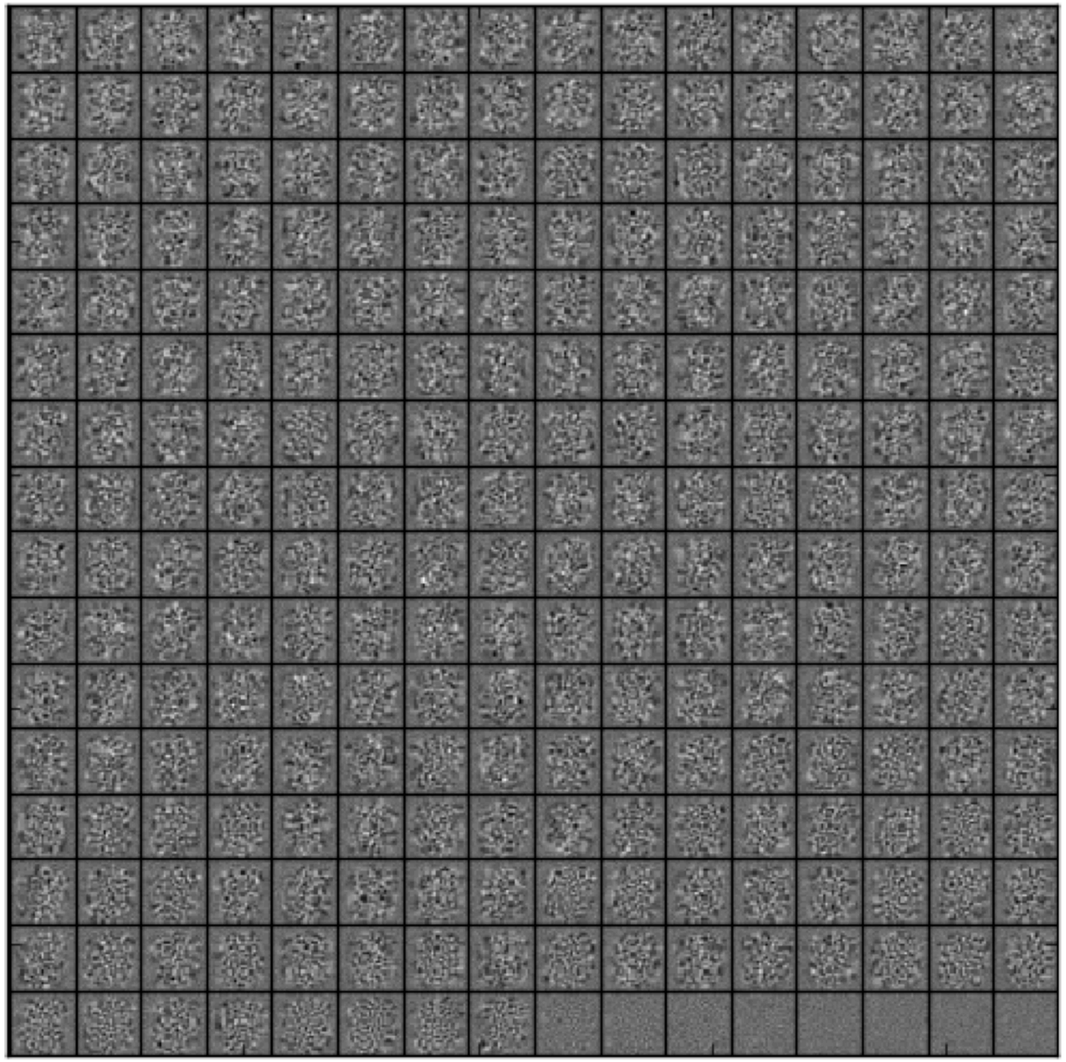
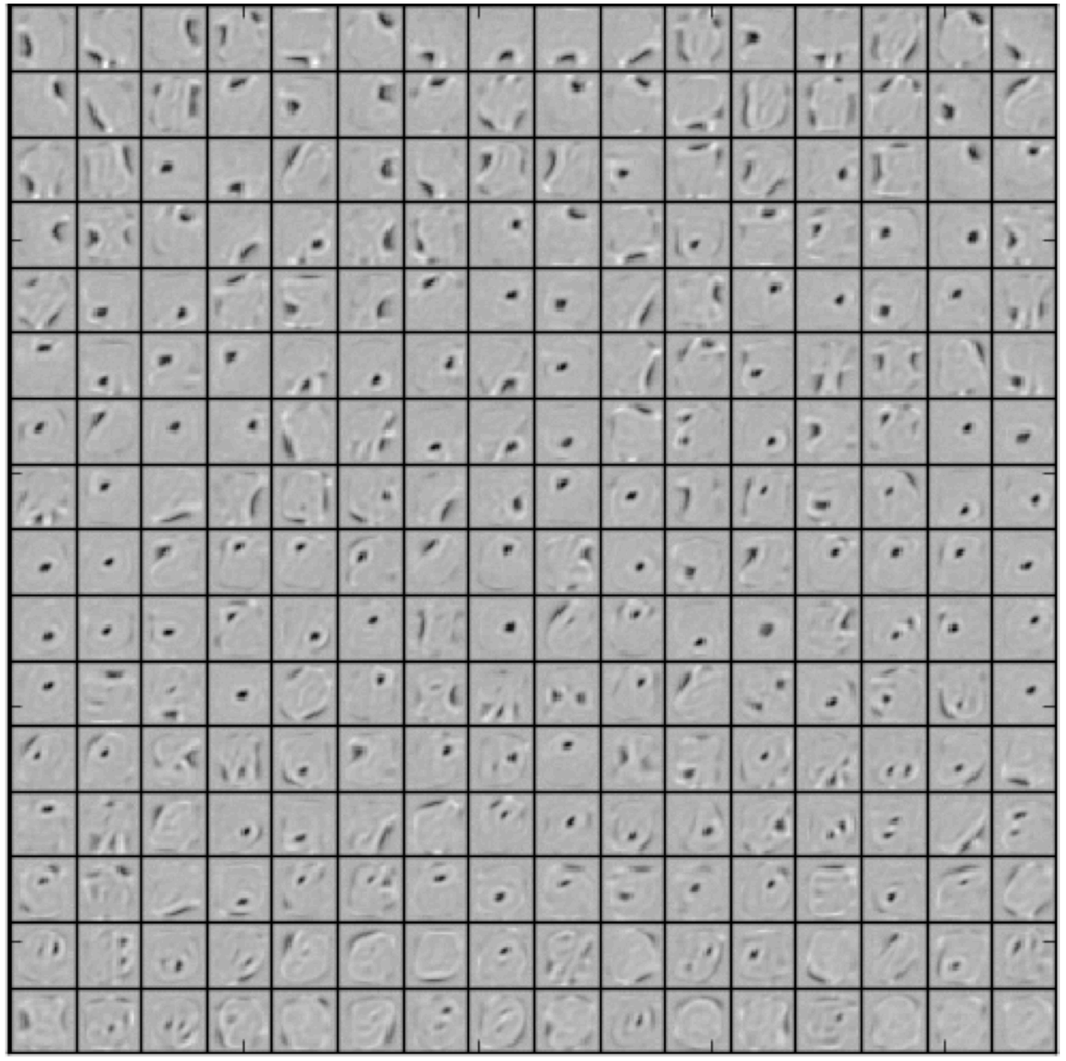


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

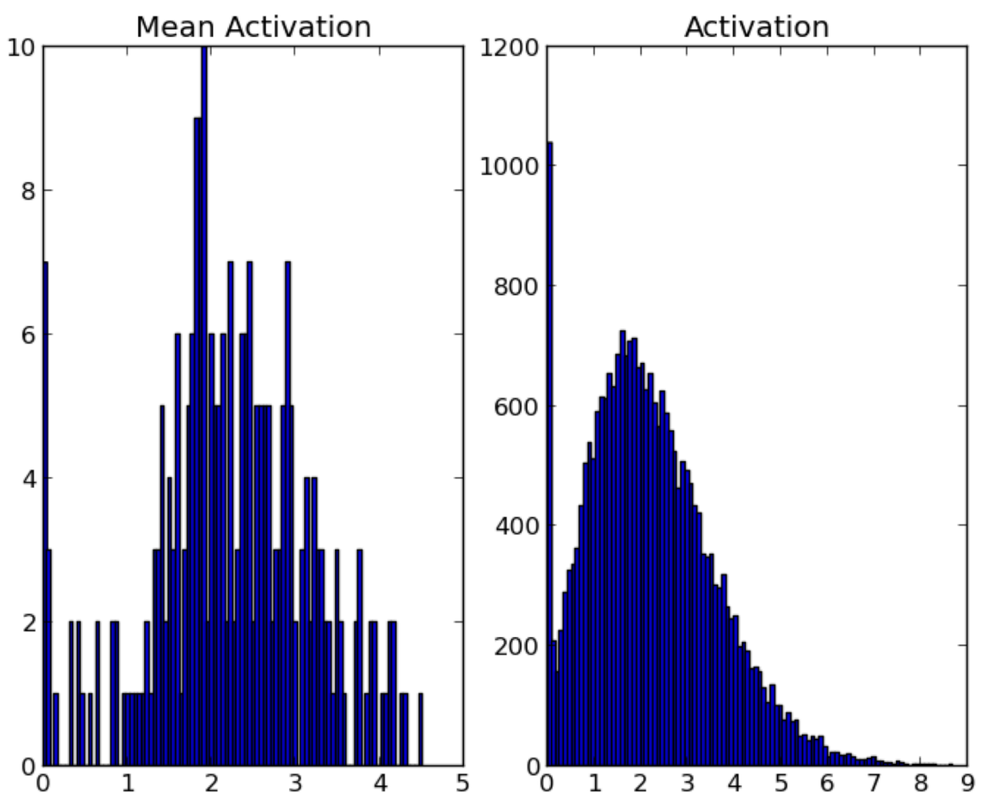


(a) Without dropout

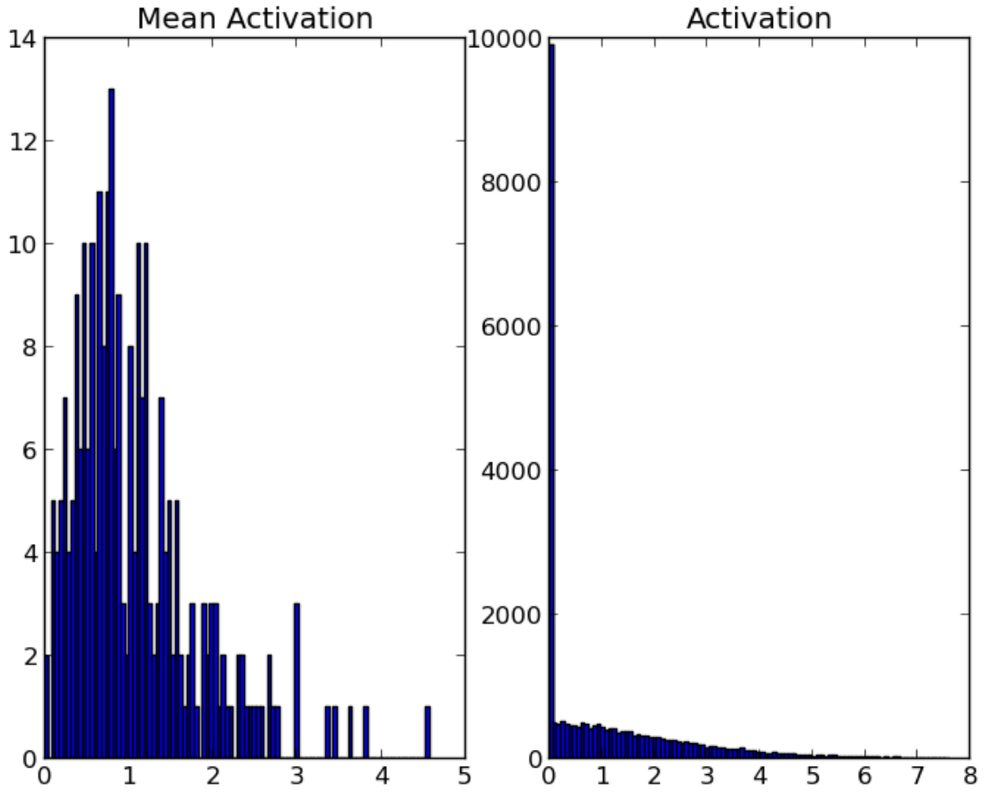


(b) Dropout with $p = 0.5$.

Figure 7: Features learned on MNIST with one hidden layer autoencoders having 256 rectified linear units.



(a) Without dropout



(b) Dropout with $p = 0.5$.

Figure 8: Effect of dropout on sparsity. ReLUs were used for both models. **Left**: The histogram of mean activations shows that most units have a mean activation of about 2.0. The histogram of activations shows a huge mode away from zero. Clearly, a large fraction of units have high activation. **Right**: The histogram of mean activations shows that most units have a smaller mean mean activation of about 0.7. The histogram of activations shows a sharp peak at zero. Very few units have high activation.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

https://arxiv.org/abs/1409.0473

NIPS (2017)