# Image Classification: Statistical Reasoning for Deep Learning Methods

Statistical interpretation of the effectiveness of some popular methods used in Deep Learning

# Agenda

1. Introduction
2. Batch Normalization
3. Dropout
4. Loss Function Selection
5. Experiment and Result
6. Conclusion
7. Q&A

# Introduction

# Introduction

- Deep Learning is ubiquitous nowadays

- but **Interpretability** of DL remains challenging

- Engineers tend to use DL models as “**magic black boxes**”

# Introduction

Models with little interpretability sabotages the reliability of the decision making process it participates, therefore, we aim to:

- Study the mathematical reasoning behind the methods used in  DL
- Reason and explain why certain DL components work in a **statistical perspective**
- Open the “black box” using statistics

# Batch Normalization technique

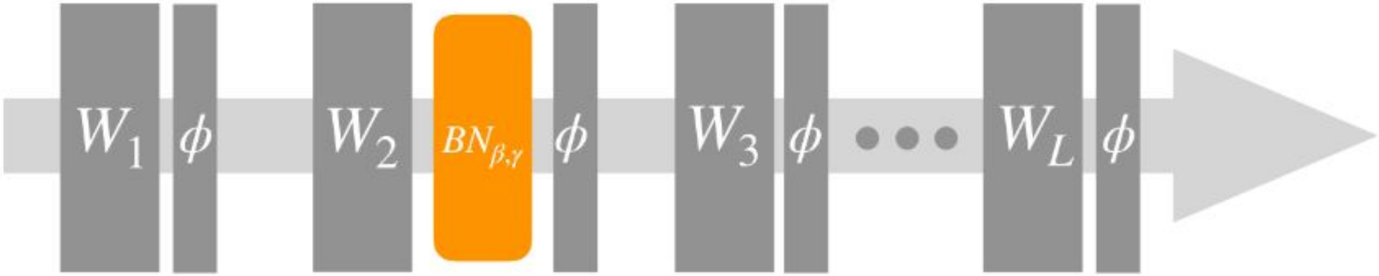
# Why Normalization ?

- Normalize the data on a similar scale to stabilize the gradient descent step and help the model converge faster.

- Batch Normalization

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

Not just to normalize before the network but also to **normalize within the network**



# Batch Normalization

- Internal Covariate Shift

  *"We refer to the change in the distributions of internal nodes of a deep network, in the course of training, as Internal Covariate Shift."* - *Ioffe and Szegedy 2015*

  Layers constantly adapt $\longrightarrow$ Slow learning

- Method

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

# Batch Normalization

- Learn parameters γ, β via backpropagation

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

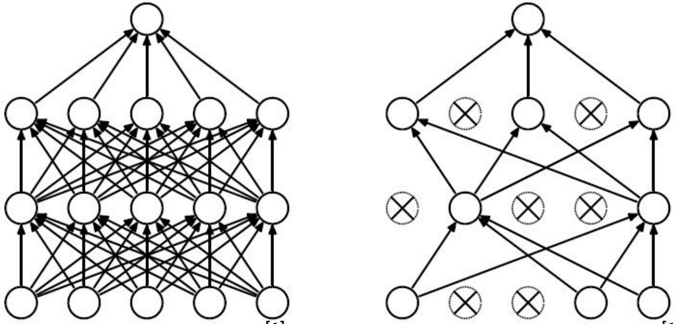
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

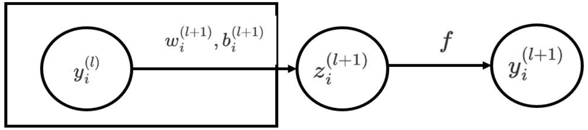
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

# Dropout

# Dropout Neurons



(a) Standard Neural Net [1]    (b) After applying dropout. [1]




$$z_i^{(l+1)} = w_i^{(l+1)} y^{(l)} + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

**Sampling**

$$r_j^l \sim Bernoulli(p) \sim \{0,1\}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \sim \{0, y^{(l)}\}$$

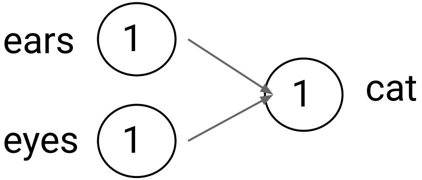
$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^{(l)} + b_i^{(l+1)}$$

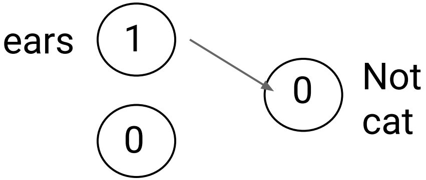
$$y_i^{(l+1)} = f(z_i^{(l+1)})$$


[1]. Dropout: A Simple Way to Prevent Neural Networks from Overfitting Journal of Machine Learning Research 15 (2014) 1929-1958

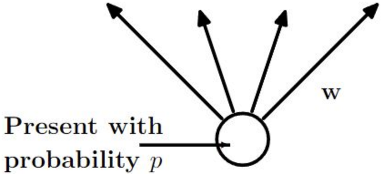
# How dropout solves overfitting

1.Avoid Co-adapting




2.Averaging




A neural net with n units, can be seen as a collection of $2^n$ possible thinned neural networks (Nitish Srivastava 2014).

[1]. Dropout: A Simple Way to Prevent Neural Networks from Overfitting Journal of Machine Learning Research 15 (2014) 1929-1958

# Bayesian Interpretation

Dropout as a Bayesian Approximation [2]

A neural network with arbitrary depth and Non-linearities + Dropout ≈ Deep Gaussian process

$$\mathcal{L}_{\text{dropout}} := \frac{1}{N}\sum_{i=1}^{N} E(\mathbf{y}_i, \widehat{\mathbf{y}}_i) + \lambda \sum_{i=1}^{L} \left( ||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2 \right).$$

$$\mathcal{L}_{\text{GP-MC}} \propto \frac{1}{N}\sum_{n=1}^{N} \frac{-\log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)}{\tau} + \sum_{i=1}^{L} \left( \frac{p_i l^2}{2\tau N} ||\mathbf{M}_i||_2^2 + \frac{l^2}{2\tau N} ||\mathbf{m}_i||_2^2 \right).$$

Setting

$$E(\mathbf{y}_n, \widehat{\mathbf{y}}(\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)) = -\log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\boldsymbol{\omega}}_n)/\tau$$

[2]. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48

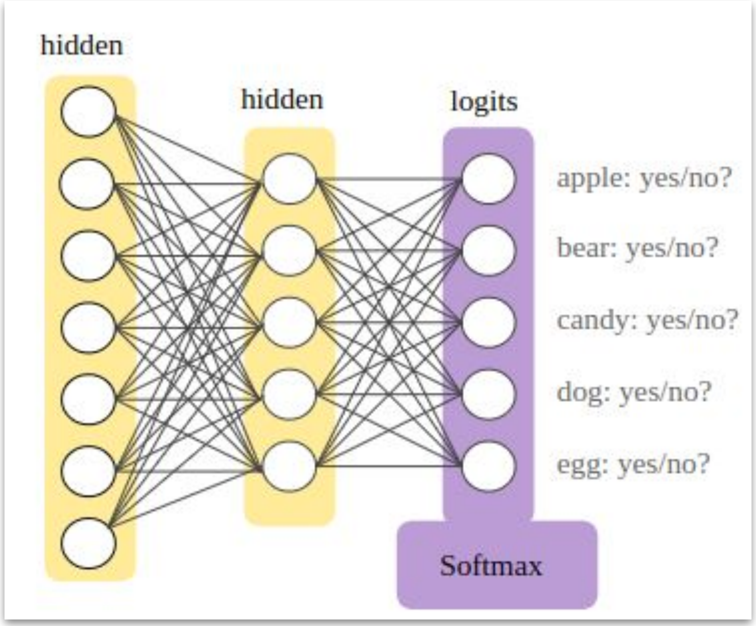
# Loss Function Selection

# Purpose

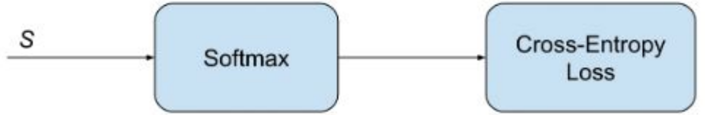
1. Provide static representation of how your model is performing–they're how your algorithms fit data

2. Provide the object function for model optimization

3. Since we are solving a multi-class classification problem, we choose softmax as activation function and cross-entropy for loss function

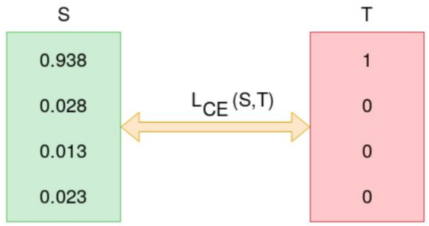


# Categorical Cross-Entropy loss




$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$




$$L_{CE} = -1\log_2(0.936) + 0 + 0 + 0$$
$$= 0.095$$

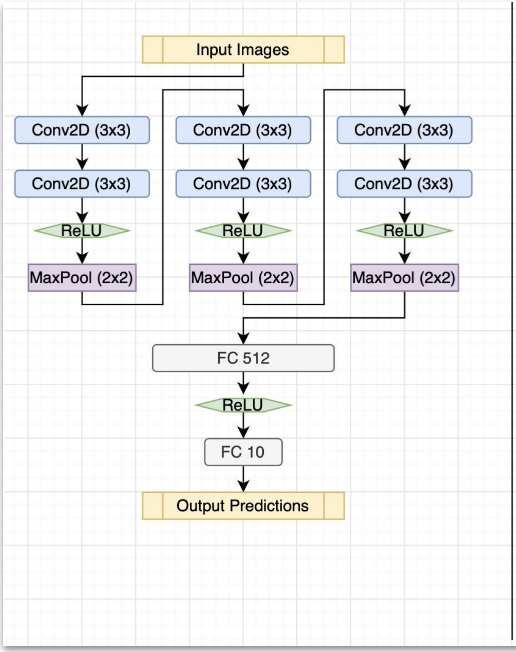
1. CNN to output a probability over the C classes for each image
2. It is a **Softmax activation** plus a **Cross-Entropy loss**
3. There is only one element of the Target vector $t_i = t_p$ which is not zero

# Experiment and Result

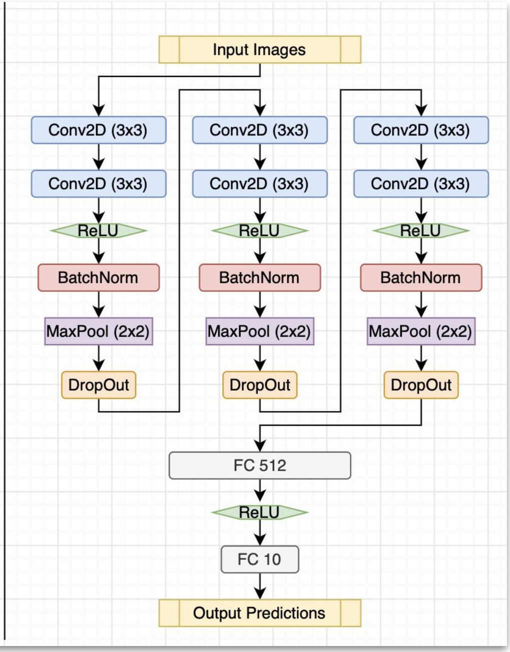
# Experiment Setup

## Dataset CIFAR-10

## Model Structure

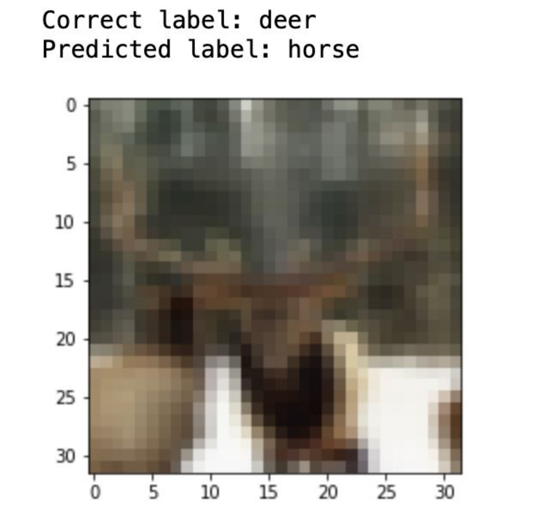



## Model Structure (Batch Norm, Dropout)

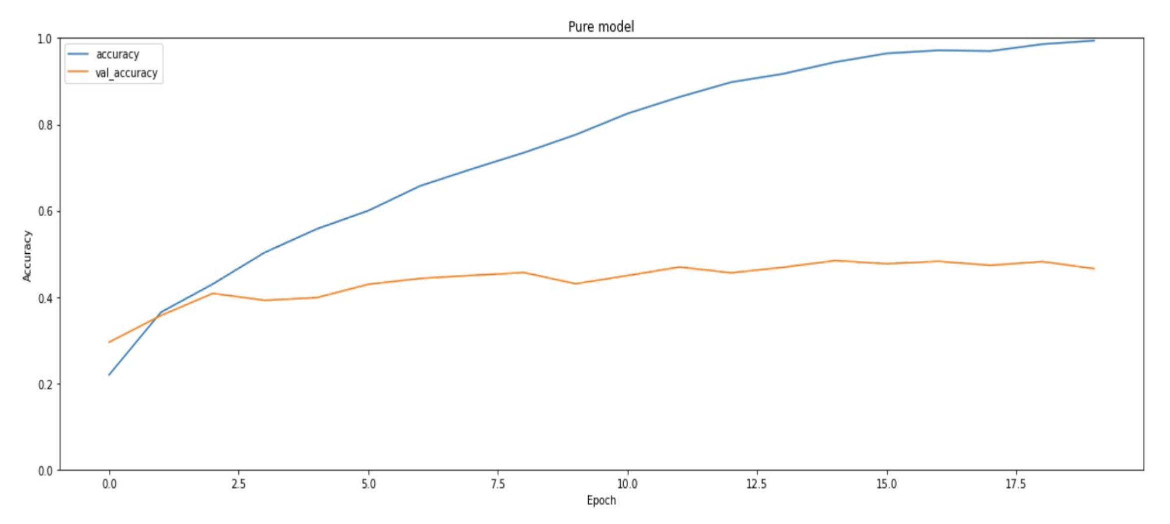


# Model Prediction

## Misclassified Example

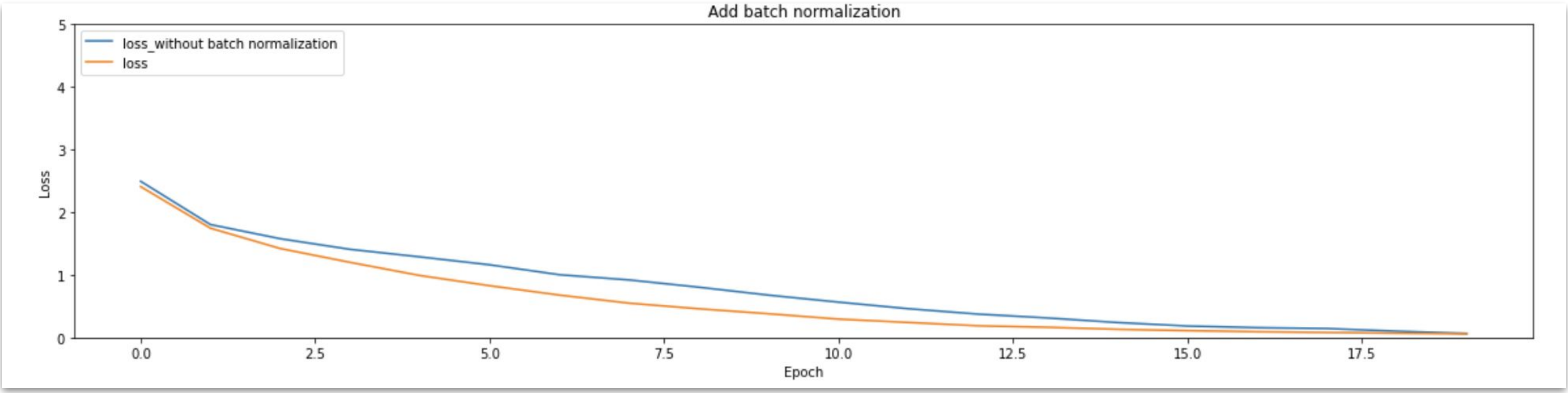



## Accuracy Graph: Training Accuracy vs Testing Accuracy



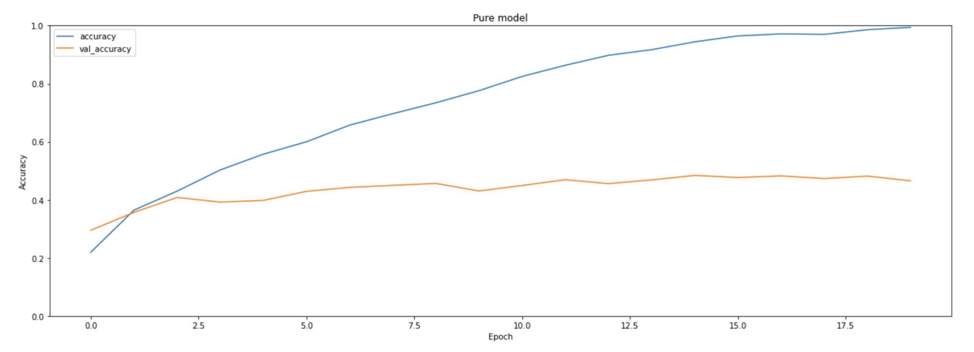
# Experiment Result

Loss without BatchNorm vs Loss with BatchNorm

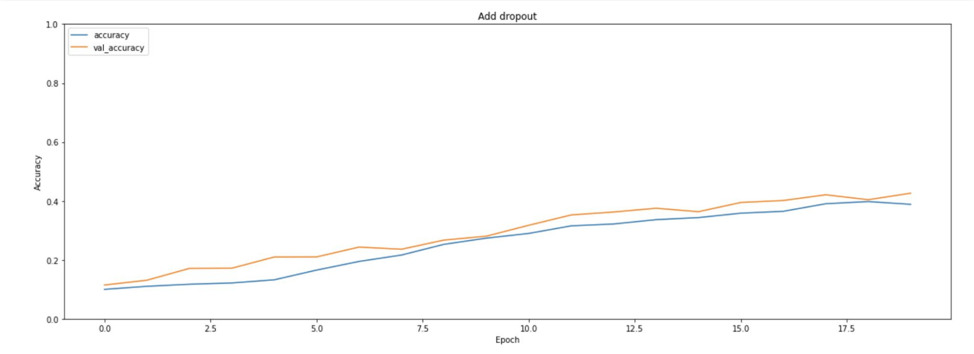


# Experiment Result

Training Accuracy and Testing Accuracy for CNN model **without Dropout**




Training Accuracy and Testing Accuracy for CNN model **with Dropout**



# Conclusion

# Conclusion

Through statistical reasoning, we saw how:

- Batch Normalization mitigates the Internal Covariate Shift which **speeds up training**
- Random Dropout served as regularization **reduces overfitting** by averaging and avoiding Co-adapting
- Minimizing the Cross Entropy loss provides a **statistical metric** to minimizing the difference between:
  - Distribution of model prediction
  - Distribution of the truth values

# Conclusion

Why DL interpretability is important:

- For science!
- Helps students learn
- Provide insights for further research and innovation
- Selling convincing stories of reliable models to investors/customers/law makers

# Q & A

# Group Members

- Hongtao Zhang, 3036047389
- Mengxuan Chen, 3036046282
- Yuexi Dong, 3036046555
- Yiming Huang, 3036101800
- Xinjing Lin, 3035572313