

# Deep Learning for Data Science

## DS 542

<https://dl4ds.github.io/fa2025/>

Data Preparation and Augmentation



# Plan for Today

- Standardizing data
- Augmenting data
- Randomizing data
- Rethinking generalization
- Random pre-training

# What is Standardization?

- Standardization is a set of preprocessing techniques to make data easier to model.
- Standardization used to be an obligatory detail in papers about neural network since it often made a big difference in whether the network could successfully train.
- Basic standardization techniques are available in scikit-learn.
  - <https://scikit-learn.org/stable/modules/preprocessing.html>

# Standardizing Mean and Variance

1. Subtract out the mean of the training distribution.
2. Divide by the standard deviation of the training distribution.

Details:

- Usually doing this column/element-wise.
- For images, sometimes channel-wise. → sometimes standard values found online
- Save these training means and standard deviations to apply to other data.
  - Validation/test/production usage uses values from training!

# What Does This Standardization Do?

- Standardizing mean makes the transformed input have **mean zero**.
- Standardizing the standard deviation makes the transformed input have **standard deviation one**.
- Assuming **later inputs are from a similar distribution**.

# Why Does Standardization Help?

Think back to backpropagation...

At beginning of training,

- Forward pass values are proportional to the input values.
- Parameter gradients computed in backpropagation are multiplied by those forward values.

# Homework 3 – Logistic Regression

```
df = pd.read_csv("https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/raw/refs/heads/master/diabetes.csv")
df.head()
```

|   | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI  | DiabetesPedigreeFunction | Age | Outcome |
|---|-------------|---------|---------------|---------------|---------|------|--------------------------|-----|---------|
| 0 | 6           | 148     | 72            | 35            | 0       | 33.6 | 0.627                    | 50  | 1       |
| 1 | 1           | 85      | 66            | 29            | 0       | 26.6 | 0.351                    | 31  | 0       |
| 2 | 8           | 183     | 64            | 0             | 0       | 23.3 | 0.672                    | 32  | 1       |
| 3 | 1           | 89      | 66            | 23            | 94      | 28.1 | 0.167                    | 21  | 0       |
| 4 | 0           | 137     | 40            | 35            | 168     | 43.1 | 2.288                    | 33  | 1       |

↑  
~0-10  
big

small

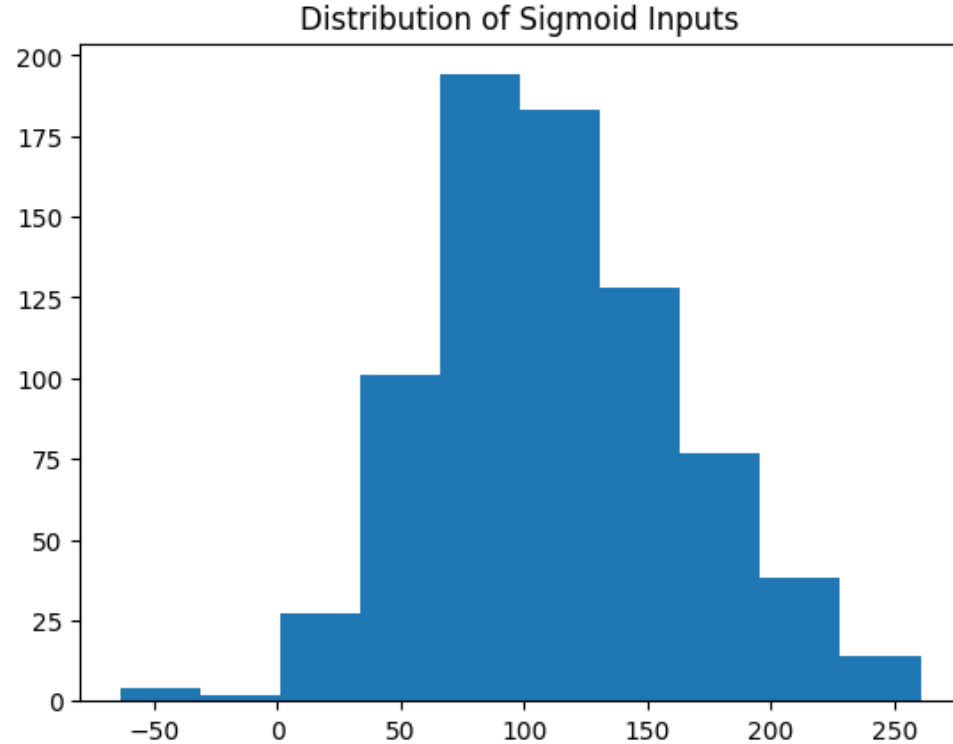
small

# Normally Distributed Initializations

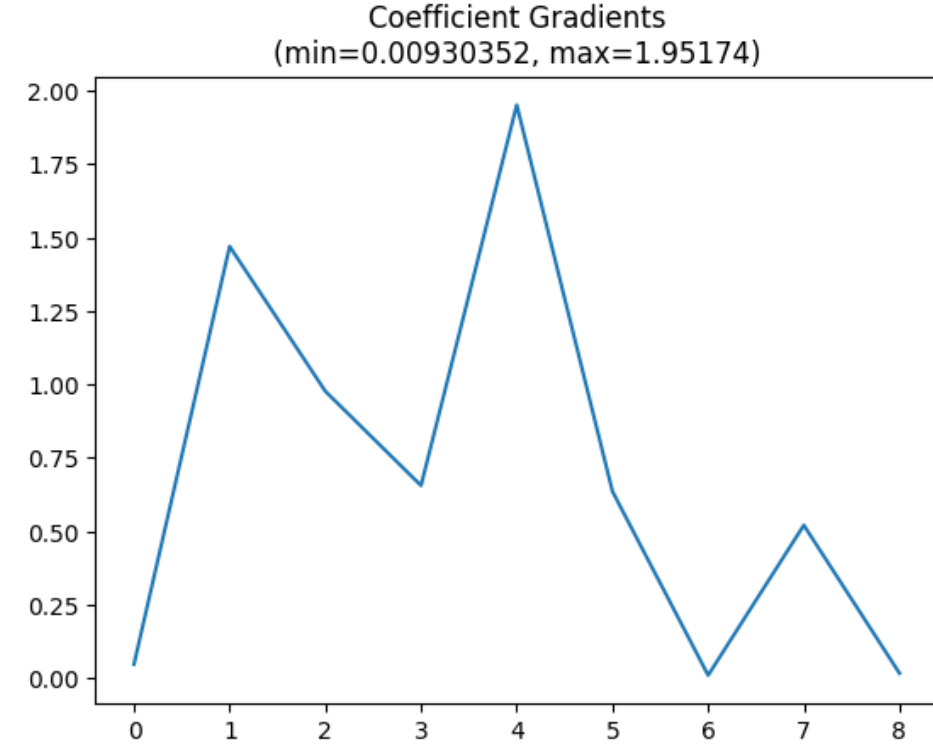
Standard normal

$\sigma(\text{linear of } x)$

## Sigmoid Inputs

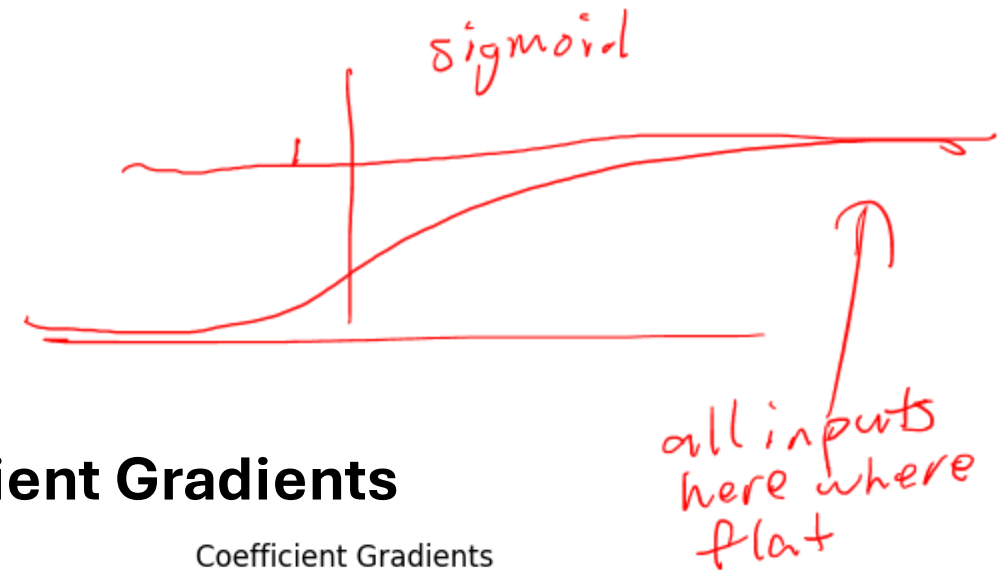


## Coefficient Gradients

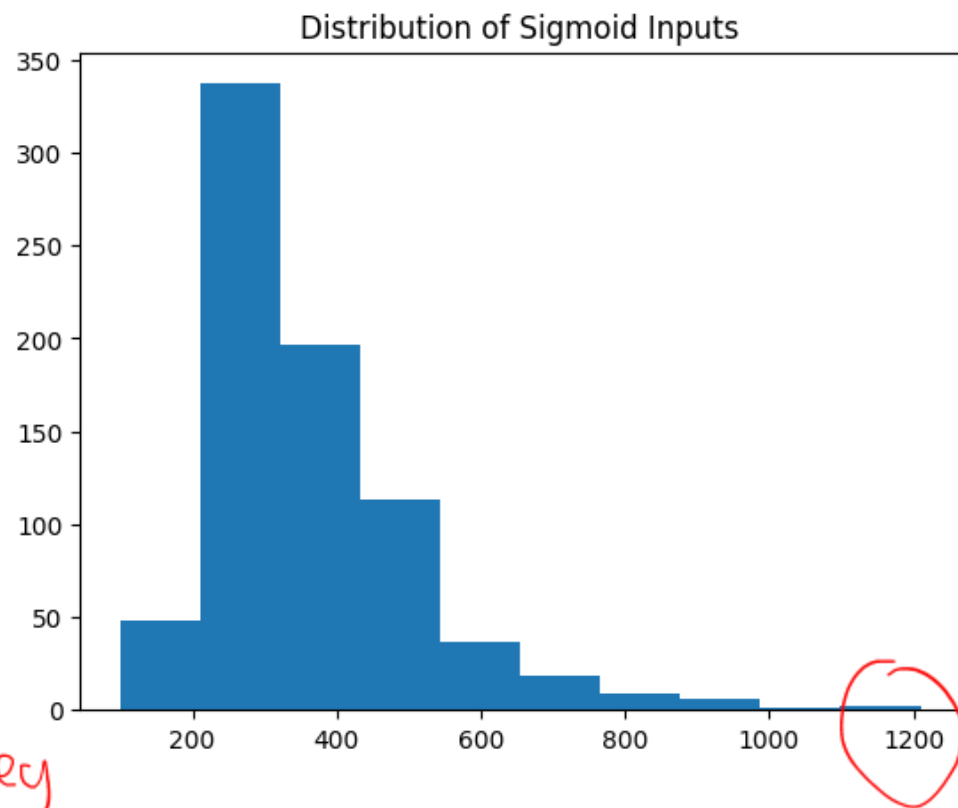




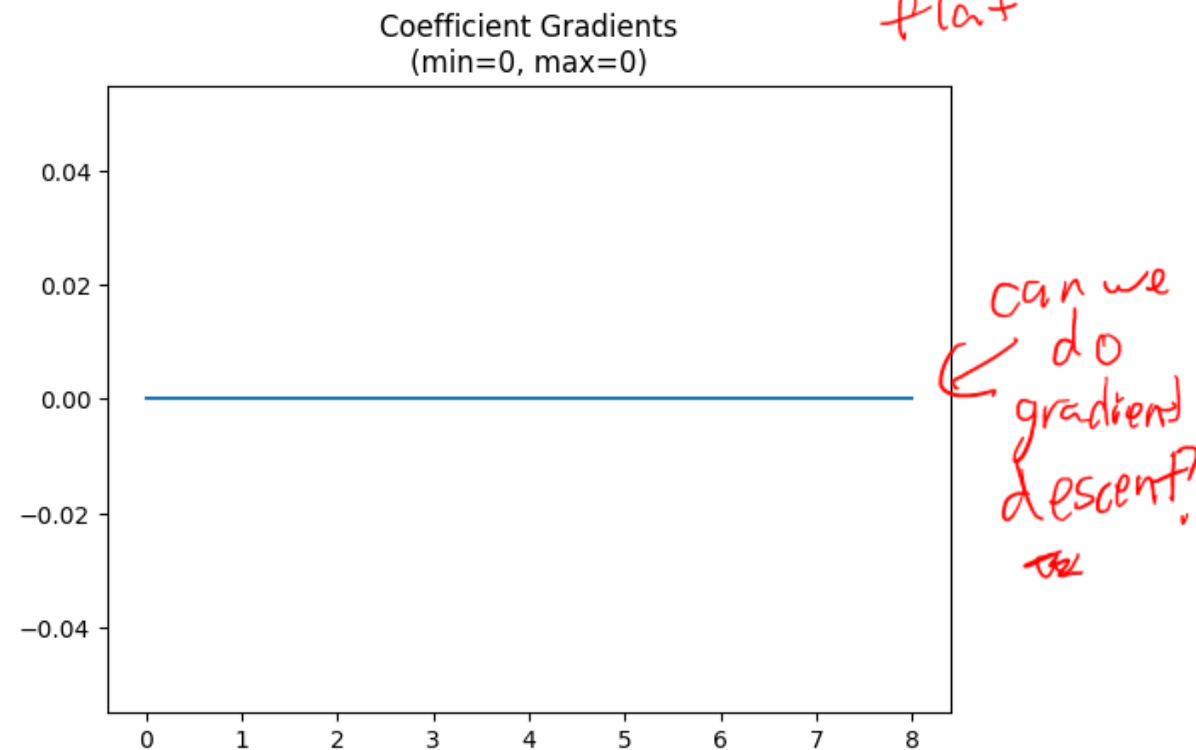
# Constant Initialization (all 1)



## Sigmoid Inputs

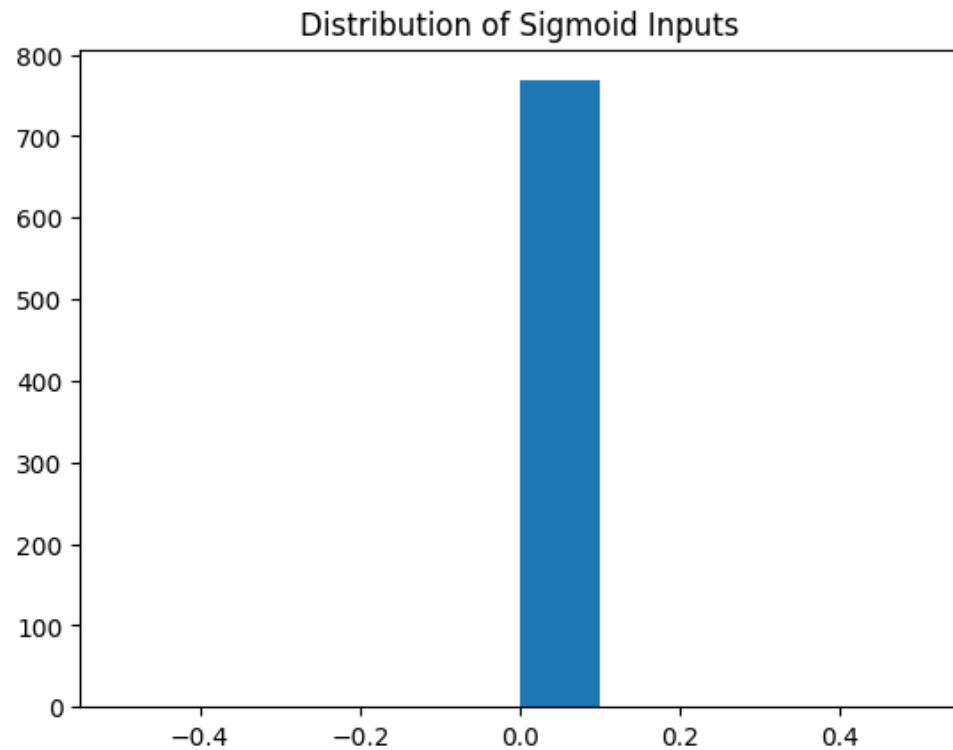


## Coefficient Gradients

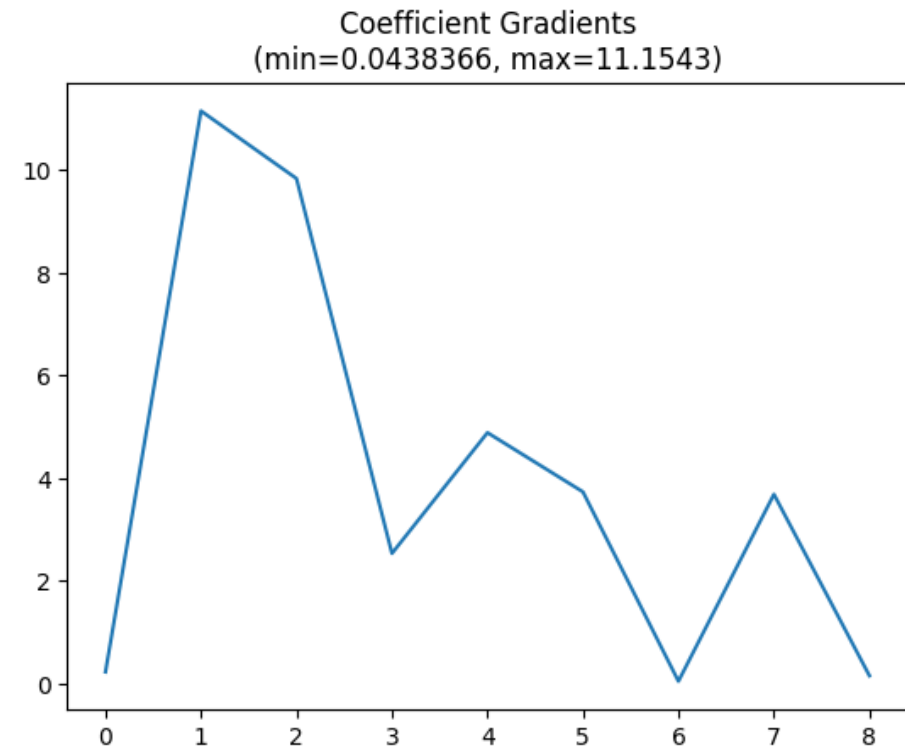


# Zero Initialization

## Sigmoid Inputs

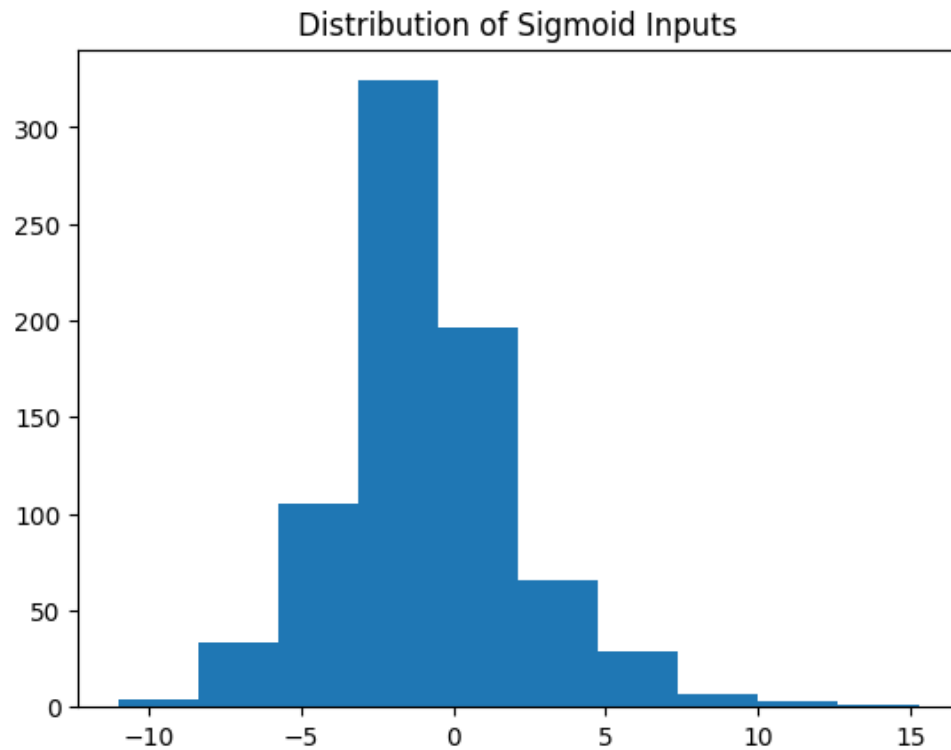


## Coefficient Gradients

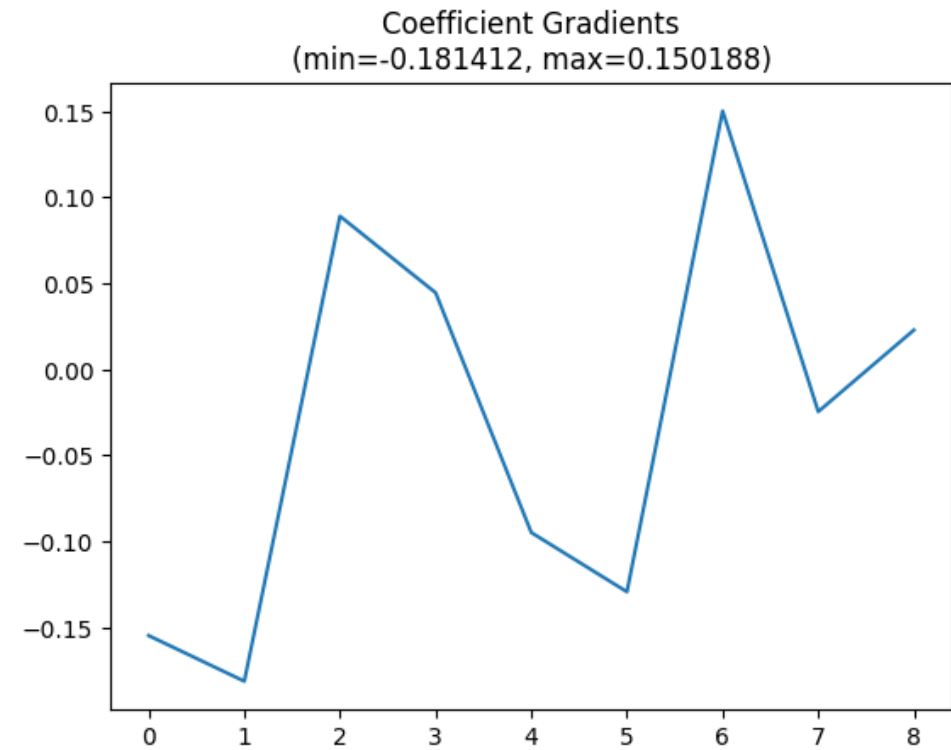


# Standardization and Normally Distributed Initializations

## Sigmoid Inputs

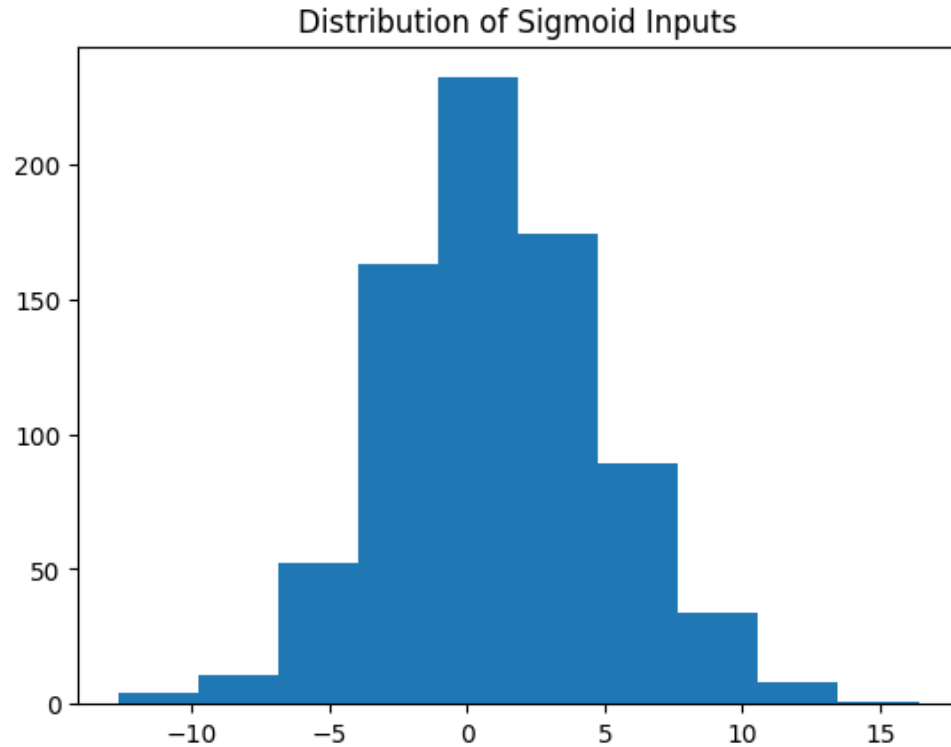


## Coefficient Gradients

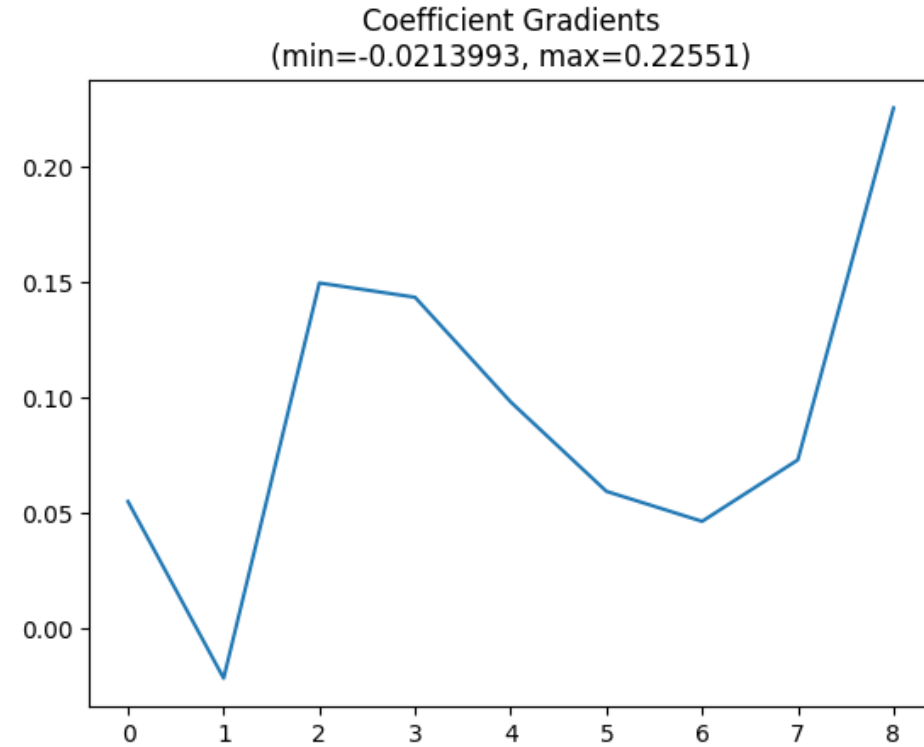


# Standardization and Constant Initializations (all 1)

## Sigmoid Inputs

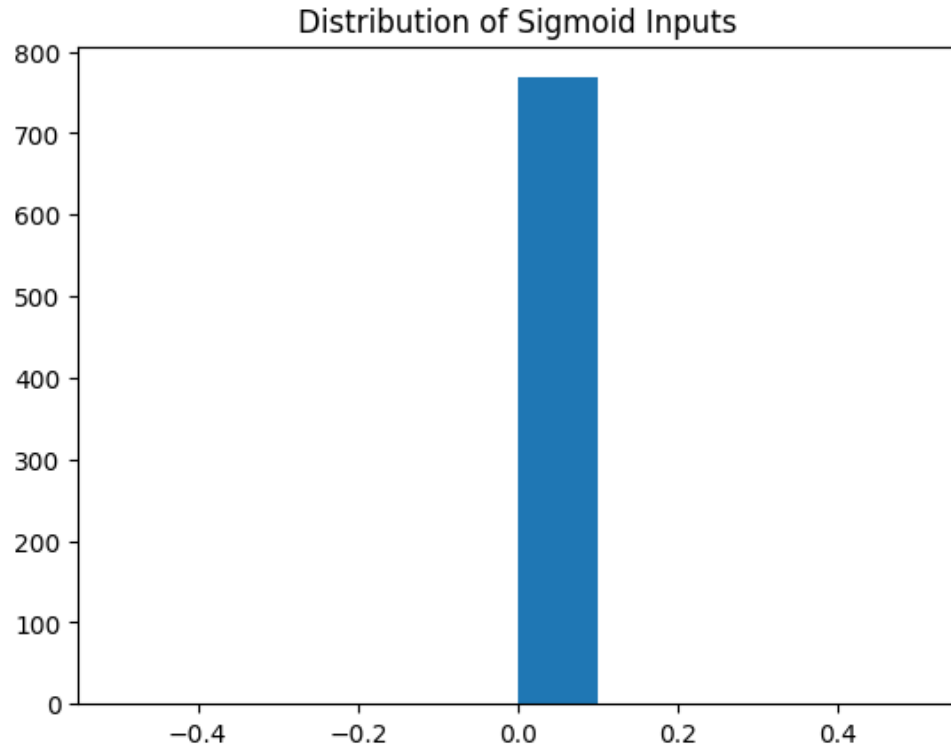


## Coefficient Gradients

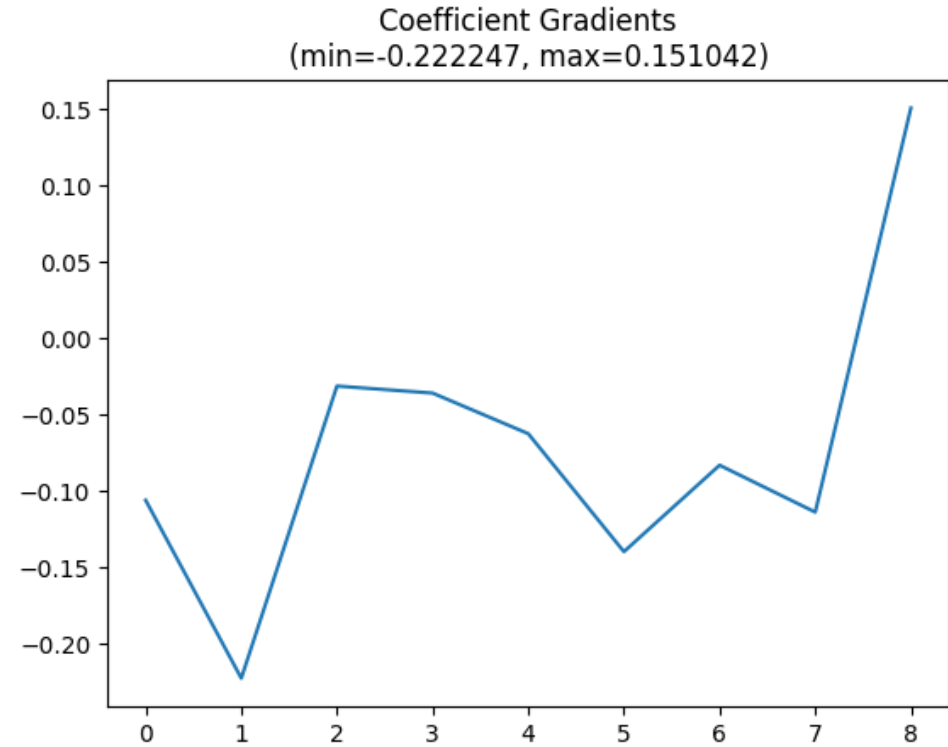


# Standardization and Zero Initializations

## Sigmoid Inputs



## Coefficient Gradients



# Whitening

- Standardizing means and variance still allows correlations between variables.
  - Covariance matrix  $\Sigma$
  - Standardizing variance gives ones along the diagonal.
  - Off-diagonal non-zero covariances are still allowed.
- Whitening is a linear transformation of the variables that removes those covariances too.
  - Removing correlations makes some modeling easier.
  - But new variables are linear combination of old variables, so slightly opaque.

Usually implemented via `sklearn.decomposition.PCA` w/whiten=True

# So why less talk about standardization?

- Still is relevant to many real-world problems.
  - Usually, data comes in **most convenient / easily checkable form**.
  - **Distinct from easiest to model form**.

Almost  
always  
should  
standardize  
for NNs

- Less important if your inputs start in a known small range.

- Images with pixel values mapped from **[0,255]** to **[0.0,1.0]**.   
 *harder to train* → *do standardize, but less benefit from [0,1]*
- Language models with input tokens **one-hot encoded** and run through **feature embeddings**.   
 *Almost all zeros, mean ~ 0*
  - Language models **would be a lot harder** if they took in integer token ids as inputs.

# Any Questions?

???

**Moving on**

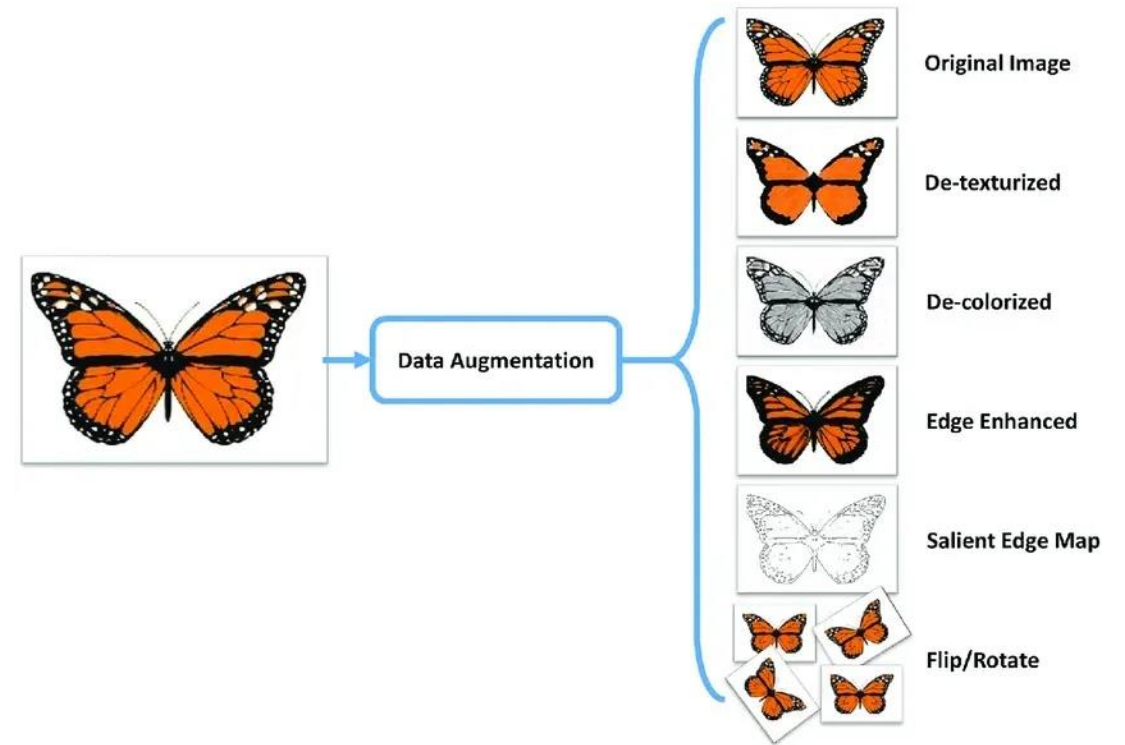
- Standardizing data
- **Augmenting data**
- Randomizing data
- Rethinking generalization
- Random pre-training



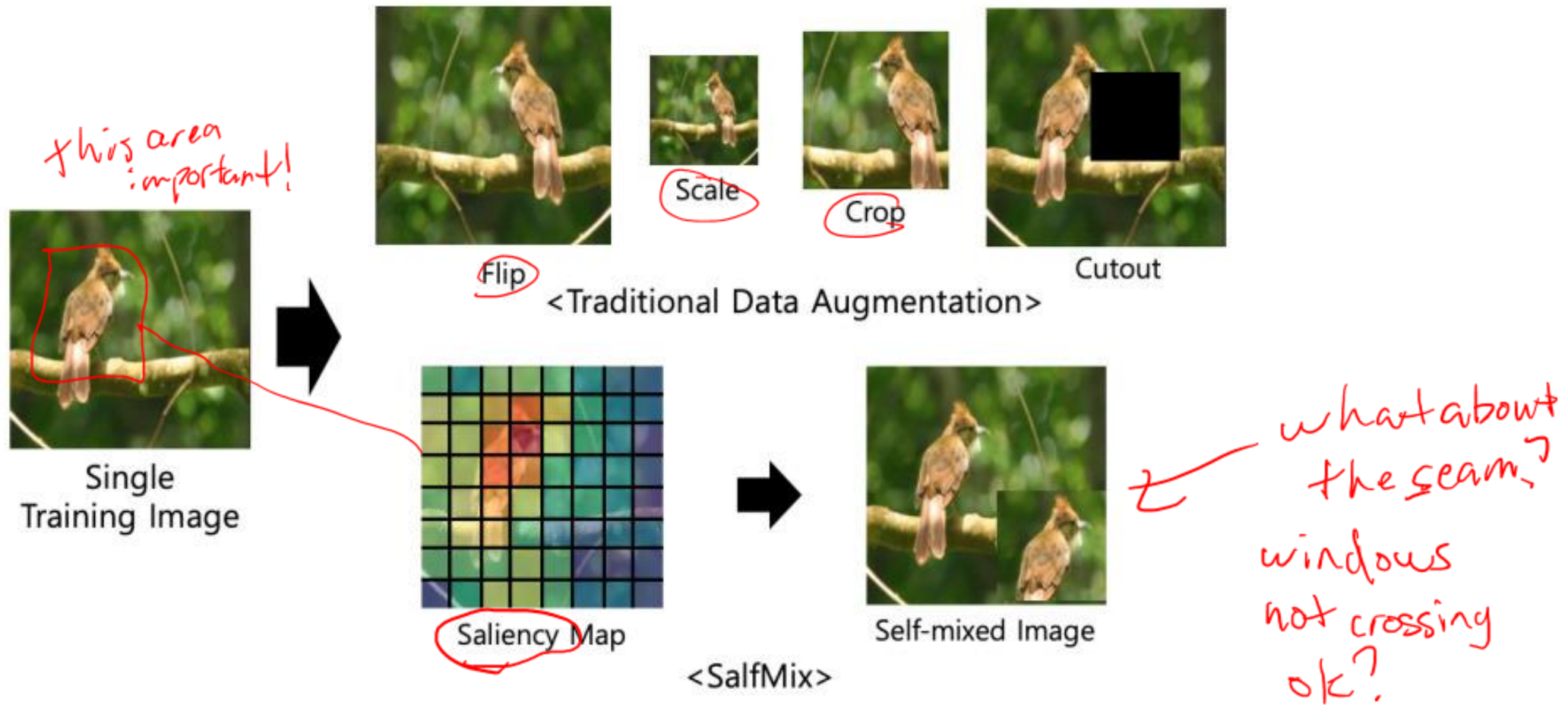
# Data Augmentation

- TLDR make data set bigger by synthesizing new inputs from old inputs...




Image source:  
<https://www.labellerr.com/blog/what-is-data-augmentation-techniques-examples-benefits/>



# New Examples of Data Augmentation



# Key Properties for Data Augmentation

1. The transformations for data augmentation should not change the correct answer!
  2. The new data ideally fits within existing distribution and does not add new competing features to learn.
- For Project 1 (tree or not),
    - Horizontal flips 
    - Vertical flips 
    - Color changes 

# What if Data Augmentation Changes the Label?

???

# Any Questions?

???

**Moving on**

- Standardizing data
- Augmenting data
- Randomizing data
- Rethinking generalization
- Random pre-training

# Randomization in Data Augmentation

- Most data augmentation code introduces more randomization...
- You pick a set of possible transformations.
  - A random subset of transforms are applied.
  - Some transformations have built-in random choices too.
    - E.g. How many pixels to shift an image.
- Effectively infinite set of augmented images?
  - But shared structure among them, so not quite as good.

# Randomization in Training Labels?

???

This was done in Project Z (butterflies)

→ Why: regularization in Cross Entropy loss

# Classification Probabilities after Randomizing Training Labels

If you deliberately change 1% of training labels,

- What is the highest probability that should be predicted?

99%

- What is the lowest probability that should be predicted?

$\frac{1\%}{(\text{classes}-1)}$

Limited range of probabilities blocks some overfitting paths.

→ Forces alternative gradient steps over "confidence"



# Cross Entropy Loss after Randomizing Training Labels

- What did overfitting look like before randomization?

very extreme predictions 0, 100%

- Can that still happen after randomization?

no, b/c loss worse then

# Any Questions?

???

**Moving on**

- Standardizing data
- Augmenting data
- Randomizing data
- Rethinking generalization
- Random pre-training

# Understanding deep learning requires rethinking generalization (Zhang et al, 2016)

- Can deep neural networks fit random labels?

- Of course!

*b/c universal approximation*

*(not 100% incorrect)*

- How big do they need to be?

- Previously:  $nd$  parameters for  $n$  inputs and  $d$  dimensions.

- New:  $2n + d$  parameters using 2 layers and RELU. *← easy*

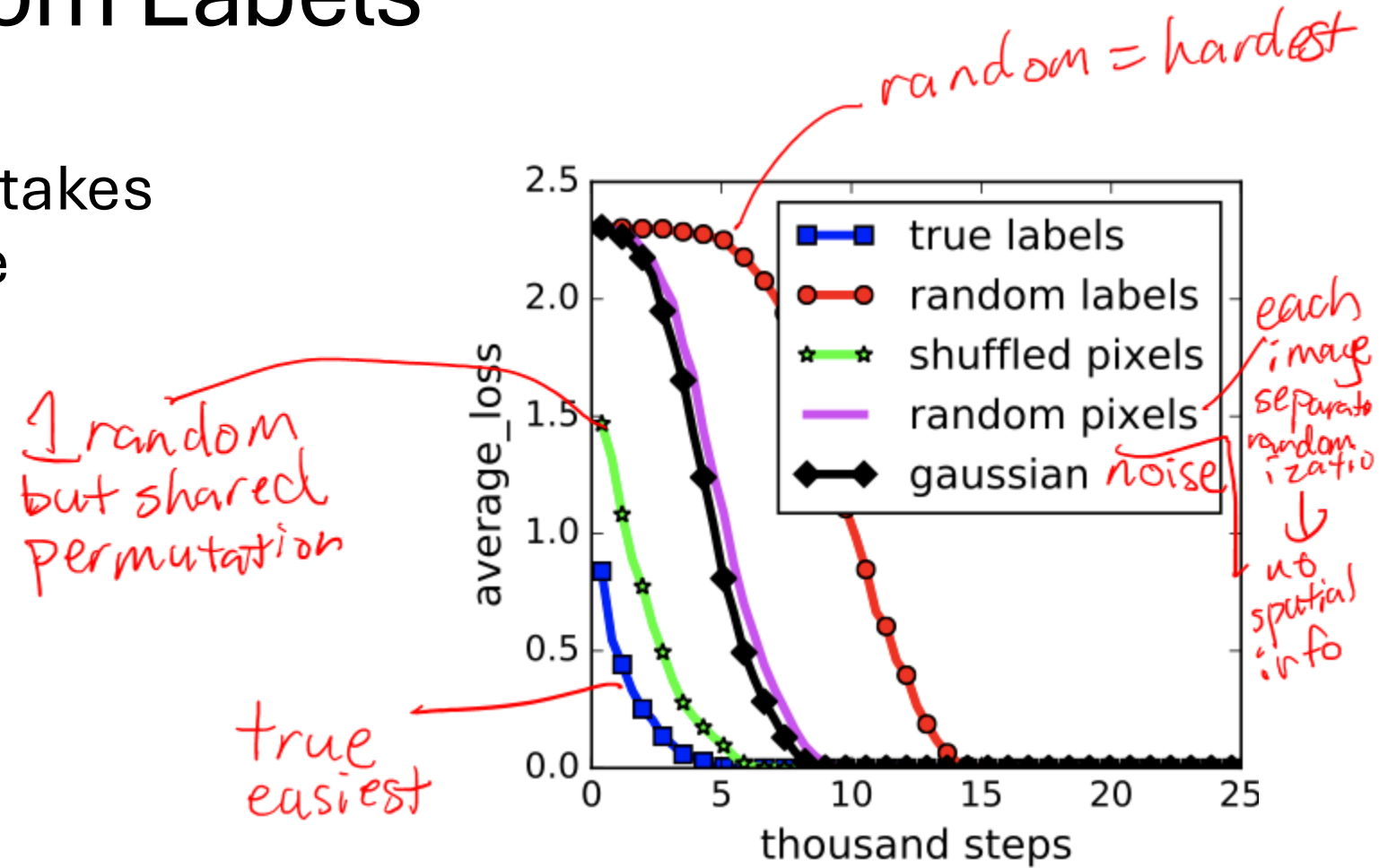
*small!*

*see appendix.*

# Fitting Pure Random Labels

- Learning random labels takes longer than learning true labels.
- In between
  - Adding noise to pixels
  - Permuting pixels

Tested on CIFAR10 and Inception architecture.

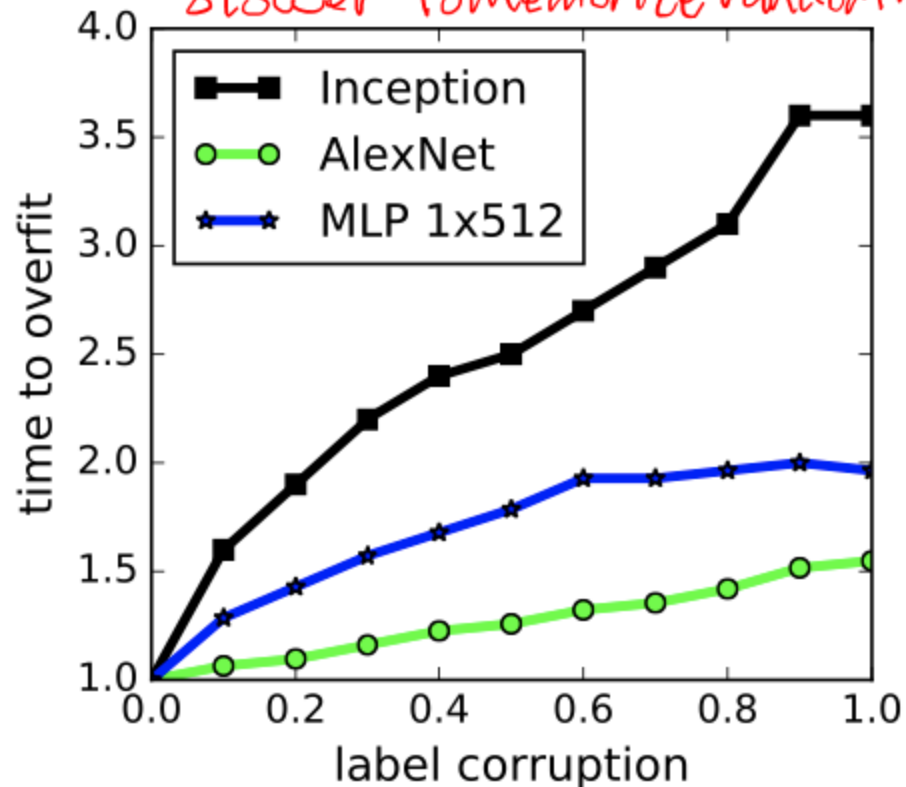


(a) learning curves

# Fitting Partially Corrupted Labels

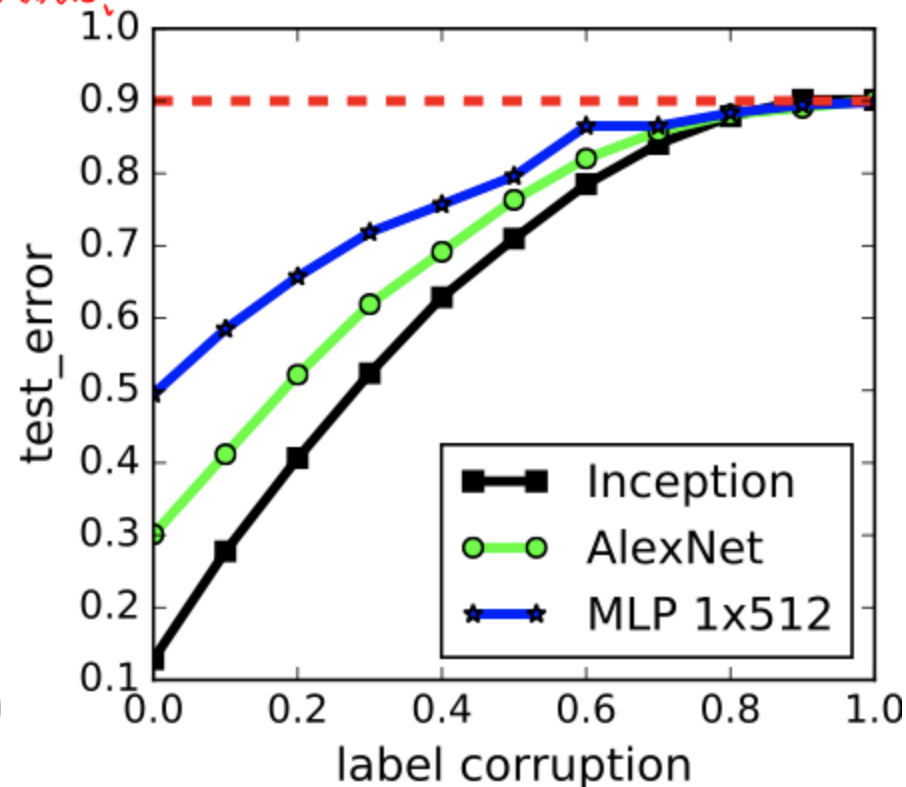
→ x-axis = fraction randomized

better architecture  
slower to memorize random labels?



(b) convergence slowdown

~~see~~ better arch resists bad labels more?



(c) generalization error growth

# Any Questions?

???

## Moving on

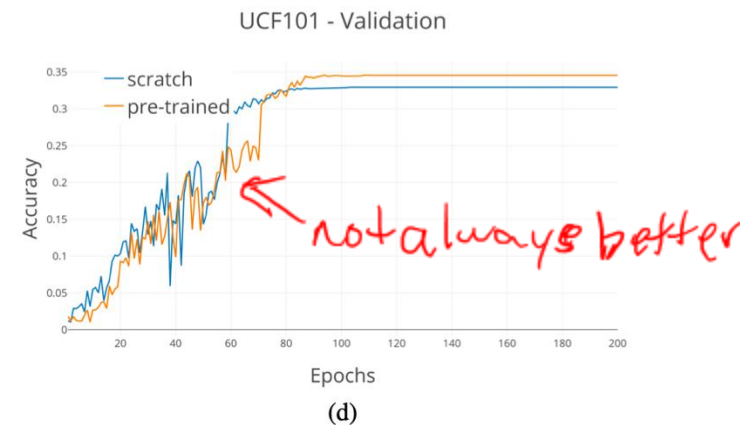
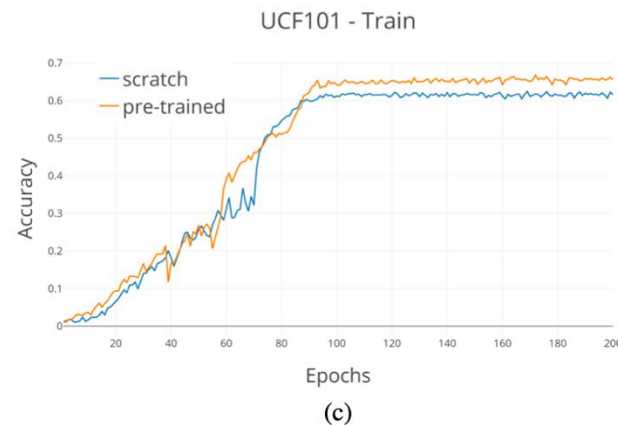
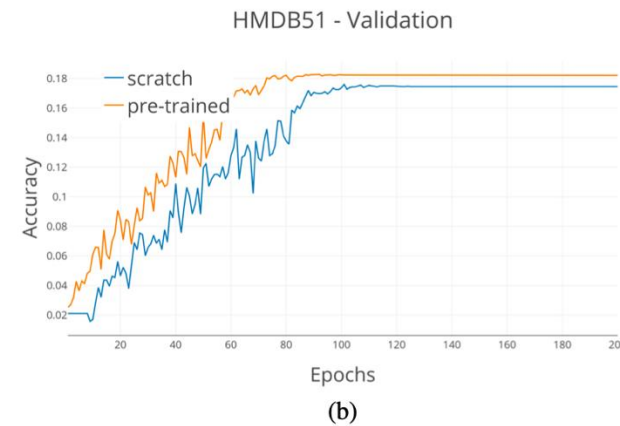
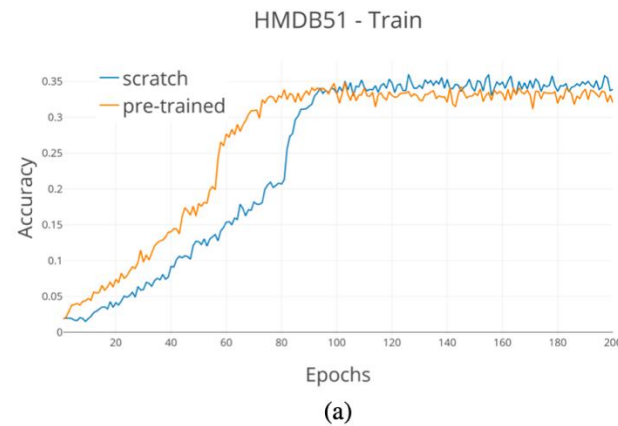
- Standardizing data
- Augmenting data
- Randomizing data
- Rethinking generalization
- Random pre-training

# Leveraging Random Label Memorization for Unsupervised Pre-Training (Pondenkandath et al, 2018)

- What if we pre-train on random labels first, then real labels?
- Why would we do that?

Hope to get a better model?

# Faster Training and Higher Test accuracy? *sometimes*



Leveraging Random Label Memorization for Unsupervised Pre-Training (Pondenkandath et al, 2018)



# Transfer Learning


- General term for leveraging training effort on one task for another task...
  - Original usage
    - Train model for one task.
    - Copy weights to new model.
    - Change output size if necessary.
    - Fine-tune for new task.
  - Previously saw this reusing pre-trained models (e.g. BERT).
- Classification*
- mostly training last layer, but usually full finetune.*
- LLM version*
- image models usually smaller than language models*

# Why Transfer Learning?

- Decrease training time for tasks with similar inputs?
- Generalize better by combining learnings across tasks?

what useful structure from  
random label pre-training?

# Why Does Transfer Learning Work with Random Labels?

- It doesn't always work. 
- But recognizing image structure seems to help distinguishing images.
  - So, pre-training on random labels **may learn useful features for distinguishing images...**

# What Do Neural Networks Learn When Trained With Random Labels? (Maennel et al, 2020)

"In this paper, we **show analytically** for convolutional and fully connected networks that **an alignment between the principal components of network parameters and data** takes place when training with random labels. We study this alignment effect by investigating neural networks pre-trained on randomly labelled image data and subsequently fine-tuned on disjoint datasets with random or real labels. We show how this alignment produces a positive transfer: **networks pre-trained with random labels train faster downstream compared to training from scratch** even after accounting for simple effects, such as weight scaling. We analyze how **competing effects, such as specialization at later layers, may hide the positive transfer.**"

# So, It Depends

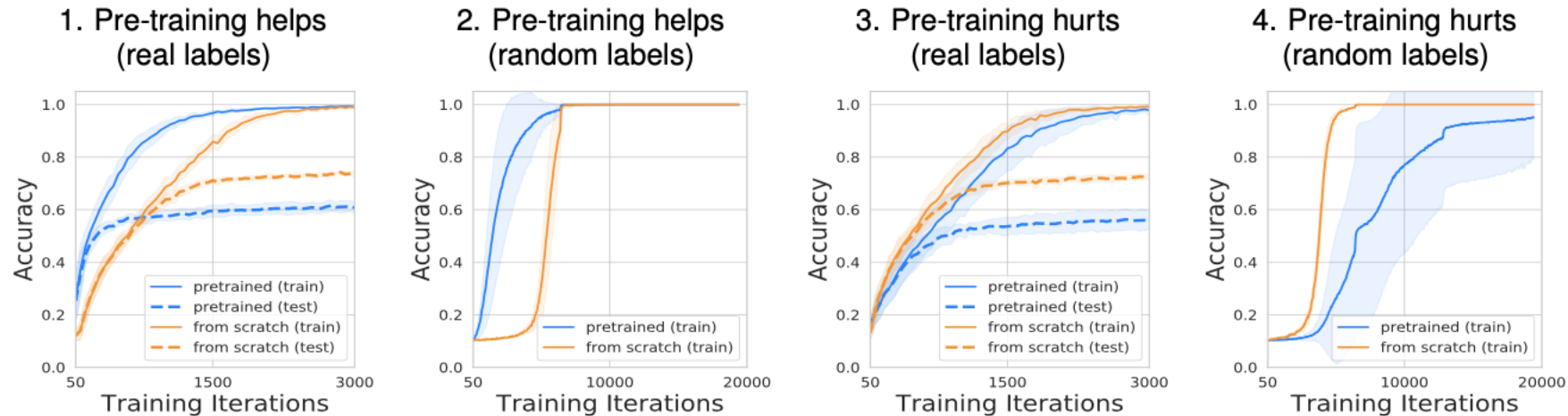


Figure 1: Pre-training on random labels may exhibit both positive (1 & 2) and negative (3 & 4) effects on the downstream fine-tuning depending on the setup. VGG16 models are pre-trained on CIFAR10 examples with random labels and subsequently fine-tuned on the fresh CIFAR10 examples with either real labels (1 & 3) or 10 random labels (2 & 4) using different hyperparameters.

What Do Neural Networks Learn When Trained With Random Labels? (Maennel et al, 2020)

# Interesting Tidbits

- They were able to reproduce the performance the random initialization performance by approximating the principal components with sampling...
  - This worked over multiple layers too.
- One mechanism that they noted hurting transfers was inactive (dead) ReLUs.

→ common in general,  
still common w/ random labels.

↓  
stuck w/ negative  
inputs + zero outputs

So stuck dead before training on real labels.

# Any Questions?

???

- Standardizing data
- Augmenting data
- Randomizing data
- Rethinking generalization
- Random pre-training