Understanding neural network training from the perspective of feature learning

Ruoqi Shen

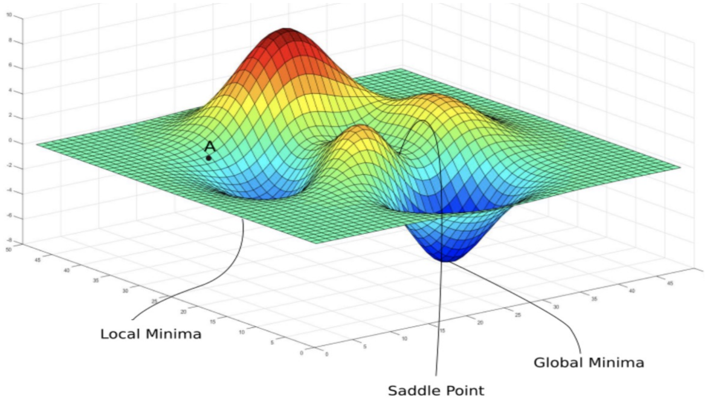
University of Washington

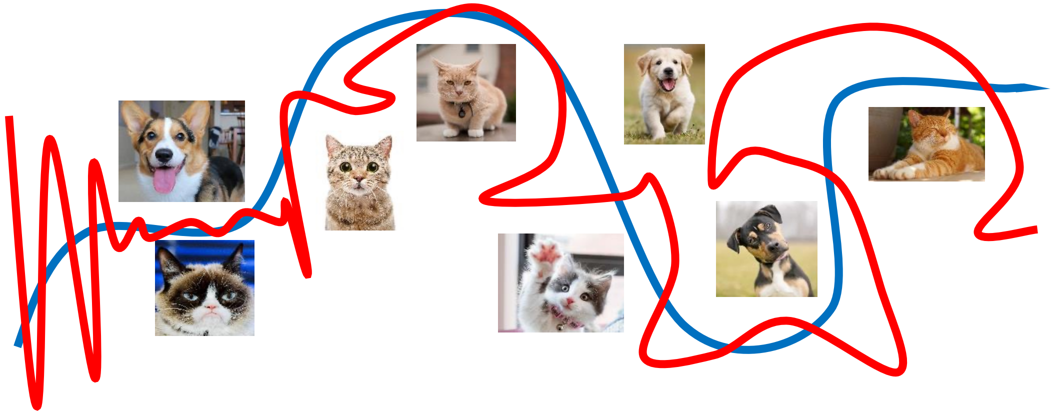
Sébastien Bubeck (MSR)

Suriya Gunasekar (MSR)

Ananya Kumar (Stanford)

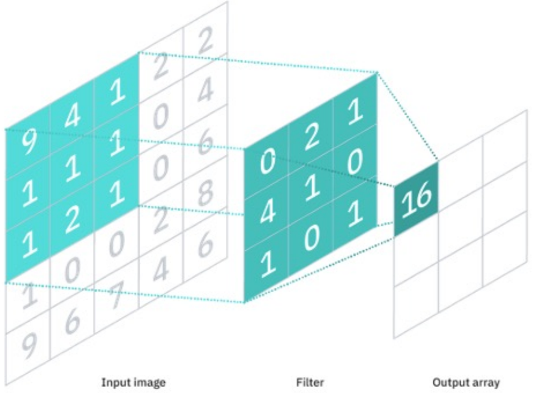
Learning & Generalization



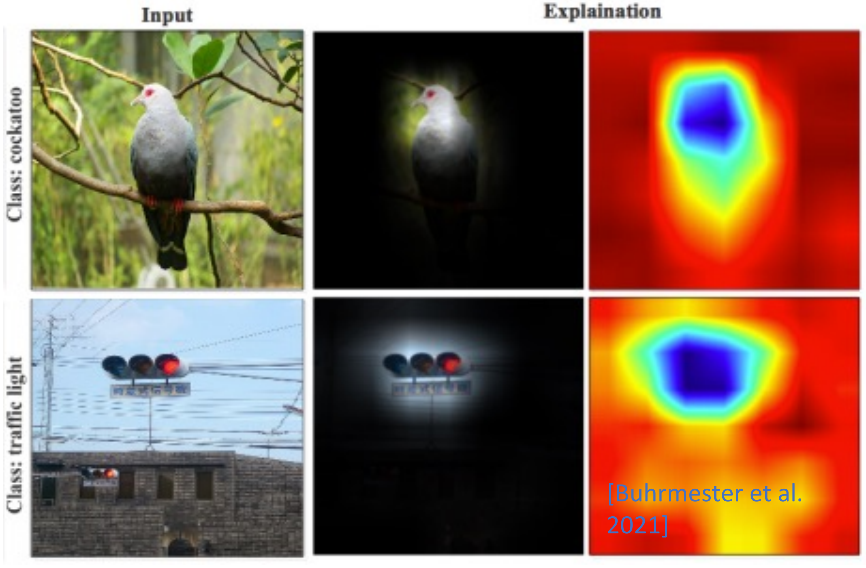


Architecture

e.g. CNN



Dataset



[Buhrmester et al. 2021]

Training Algorithm

Use
Regularization/SGD/…
to find
Flatter minima/min norm minima/….

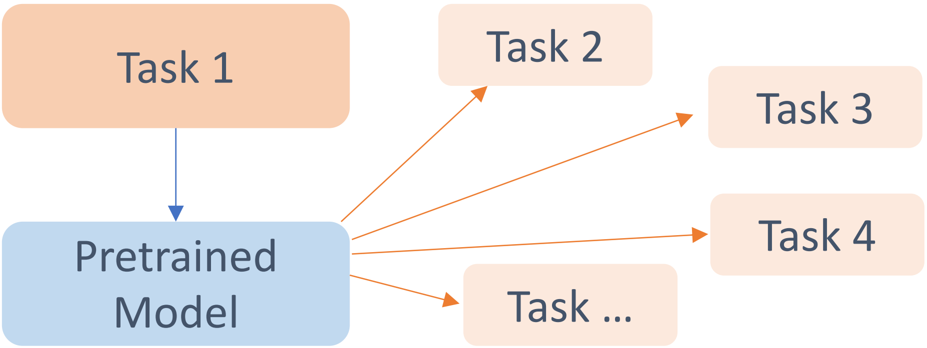
Learning & Generalization

Question: *What type of feature do neural networks prefer to learn?*

- Red / Green light?

- Generalization

- Pretraining
  - Learning features is more important than finding minima.



Outline

Understanding neural network training from the perspective of **feature learning**

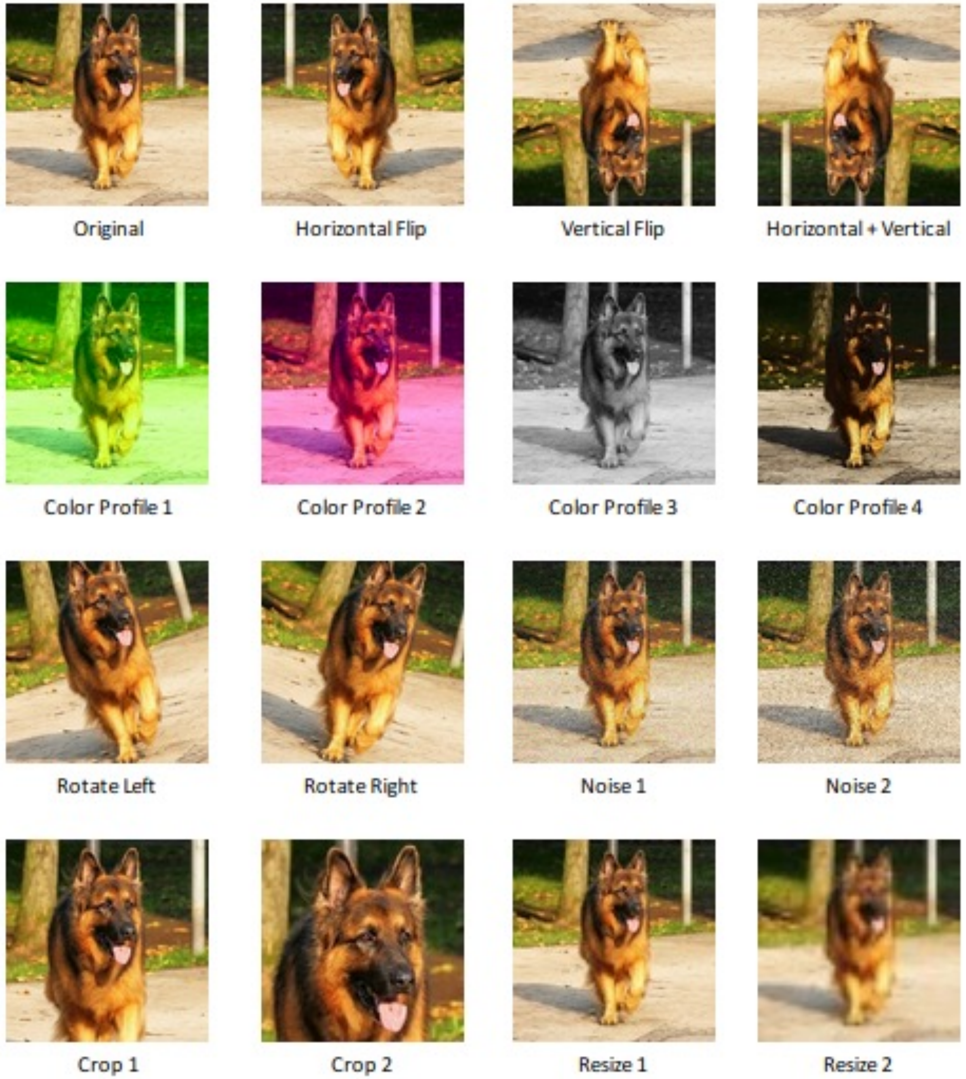
- Motivation
- Two examples:
  - How does data augmentation help supervised learning tasks?
  - How does using pretraining help training datasets with spurious correlations?

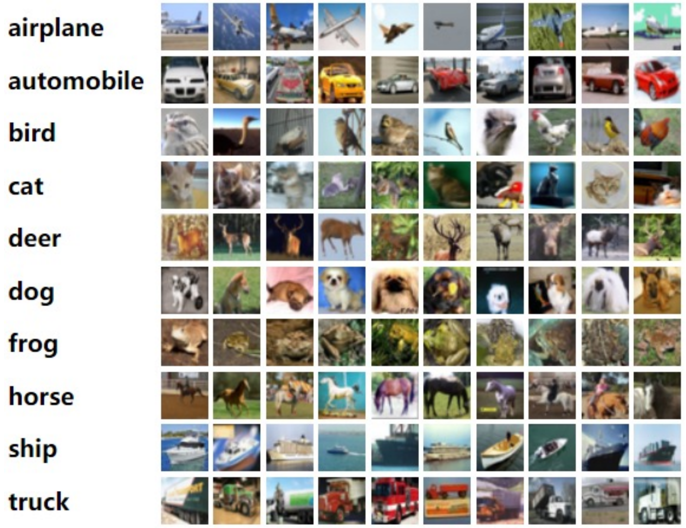
Outline

Understanding neural network training from the perspective of **feature learning**

- Motivation
- Two examples:
  - **How does data augmentation help supervised learning tasks?**
  - How does using pretraining help training datasets with spurious correlations?

Data augmentation



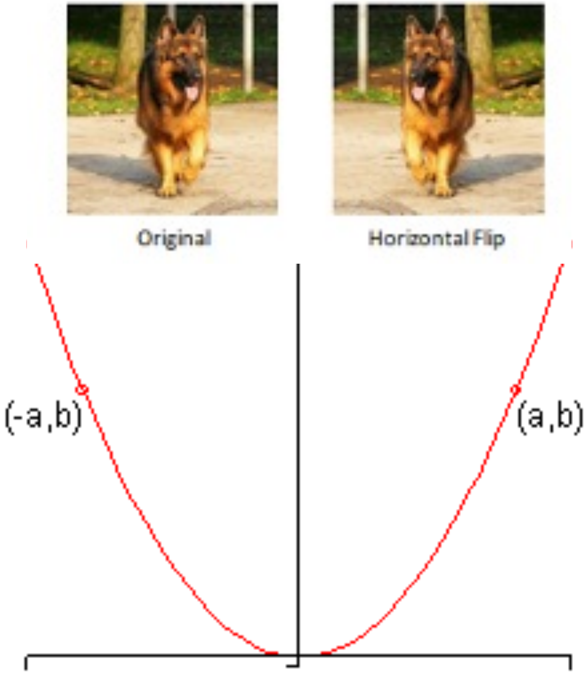


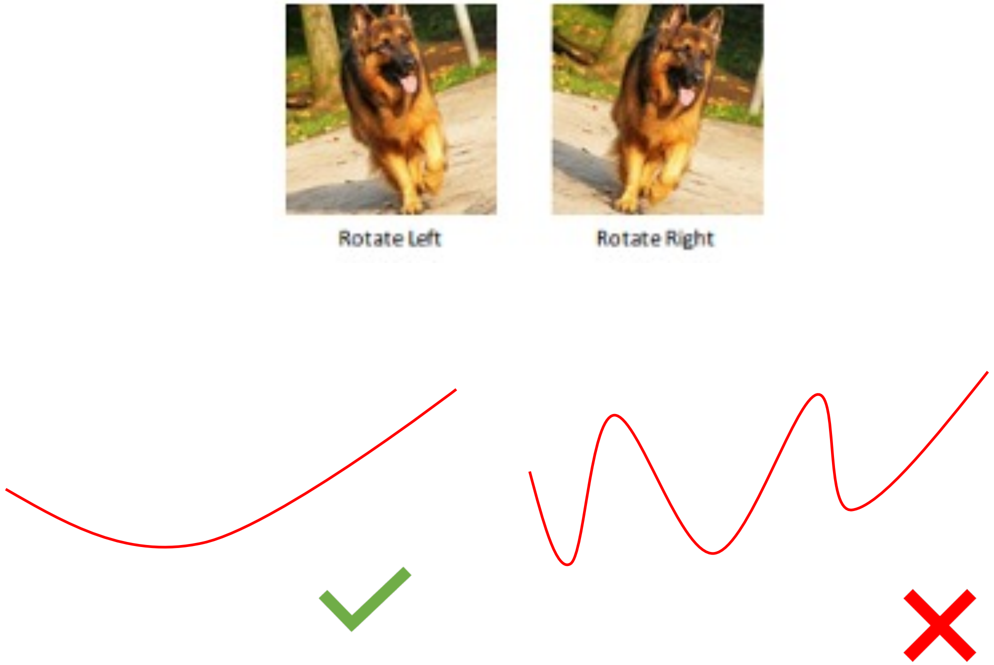
Cifar-10 Dataset

	no augmentation	basic augmentation	advanced augmentation
resnet18 (11M)	90%	96%	98%
cait_xxs36 (17M)	77%	88%	97%
vit_tiny (6M)	75%	86%	96%

Data augmentation

- **How does data augmentation help supervised learning tasks?**
  - Previous approach: Incorporate invariance? Chen, Dobriban, Lee, 2020; Mei, Misiakiewicz, Montanari, 2021

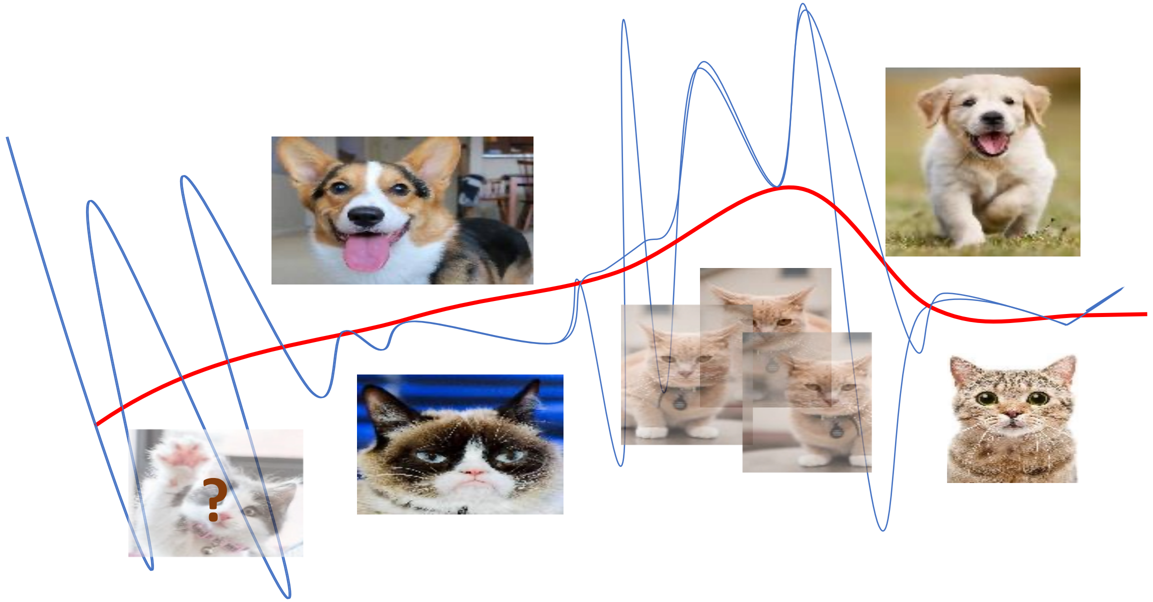




Role of data augmentation

Incorporate invariances?

- Data augmentation causes the network to learn invariance only for images that are very similar to those seen during training. Azulay and Weiss, 2019

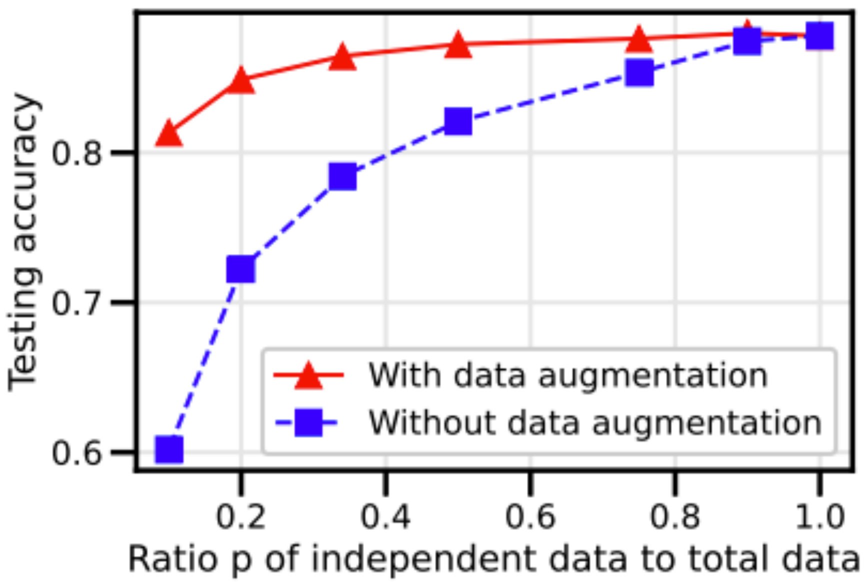


Role of data augmentation

Incorporate invariances?

Even one fixed data augmentation helps.

- Random horizontal flip + random crop (4 pixels)
- Data augmentation fixed throughout training once selected.



E.g., for ratio = 0.2,

$\mathcal{D}$ (20% of CIFAR 10 Training Set), Transformation 1 of $\mathcal{D}$, Transformation 2 of $\mathcal{D}$, Transformation 3 of $\mathcal{D}$, Transformation 4 of $\mathcal{D}$	$\mathcal{D}$ (20% of CIFAR 10 Training Set), Copy 1 of $\mathcal{D}$, Copy 2 of $\mathcal{D}$, Copy 3 of $\mathcal{D}$, Copy 4 of $\mathcal{D}$

E.g., for ratio = 0.7,

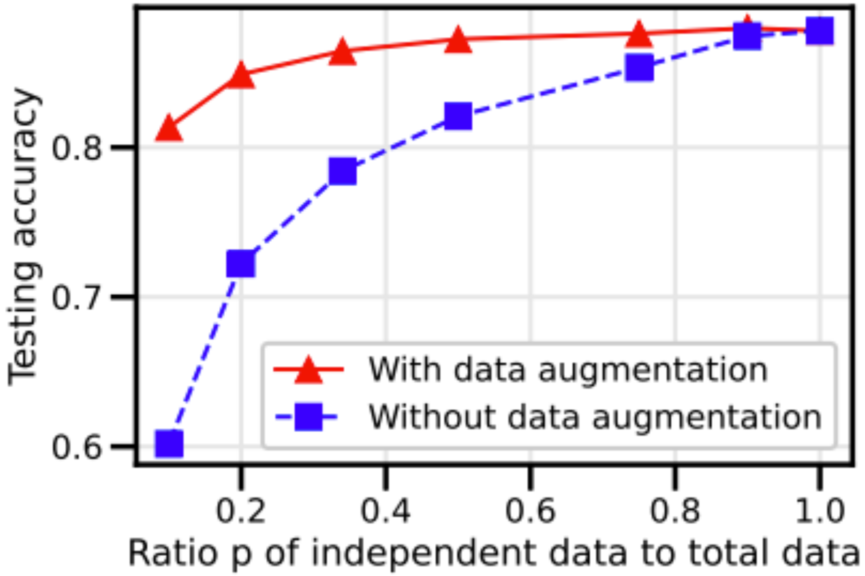
$\mathcal{D}$ (70% of CIFAR 10 Training Set), Transformation of part of $\mathcal{D}$	$\mathcal{D}$ (70% of CIFAR 10 Training Set), Copy of part of $\mathcal{D}$

Role of data augmentation

Incorporate invariances?

Even one fixed data augmentation helps.

- Random horizontal flip + random crop (4 pixels)
- Data augmentation fixed throughout training once selected.



- When $p \geq 0.5$, at most one augmented sample is seen for each sample.
- A simple data augmentation can help nearly as effectively as a new image.
- One augmented sample can only lead to very limited invariance.

Role of data augmentation

Incorporate invariances?

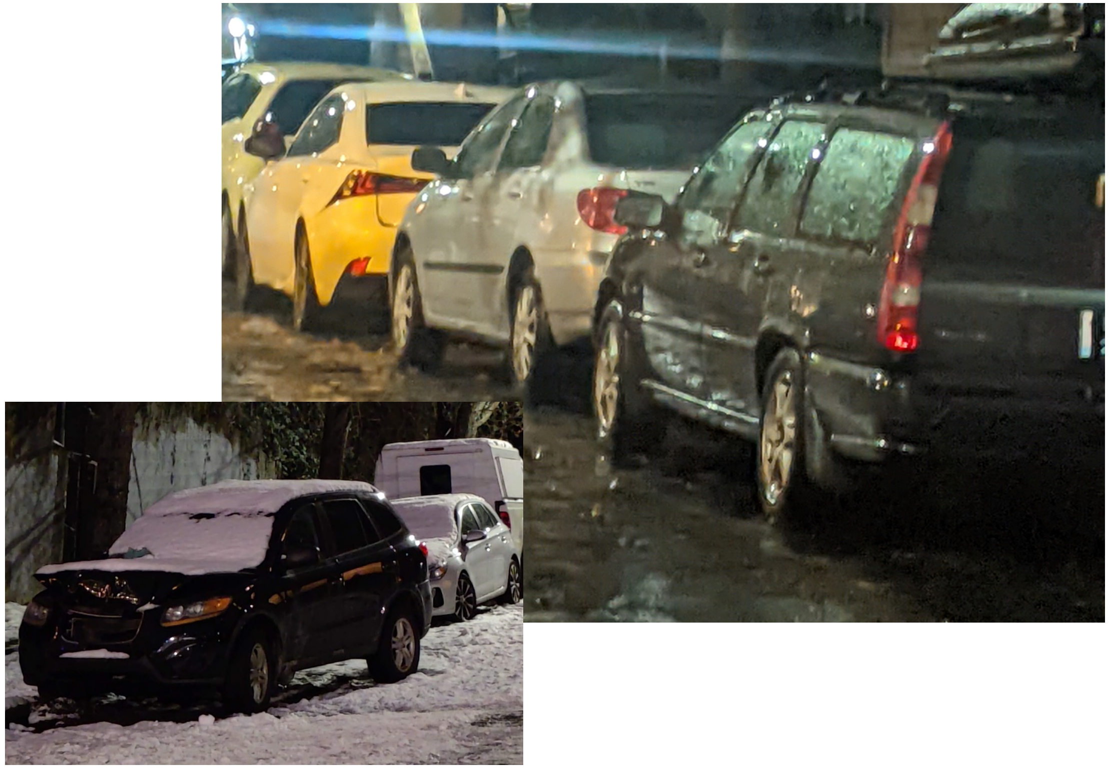
- Invariance only for images similar to those seen during training. (Azulay and Weiss, 2019)
- Even one fixed data augmentation helps.
- Not real “invariance”

Alternative explanation of data augmentation from the perspective of feature learning

- **Feature manipulation** in gradient descent dynamics

  e.g., Data augmentation increases the relative importance of “good” features compared to “bad” or “spurious” features

Feature manipulation viewpoint: bad features



Bad & Easy features: spurious feature/large noise

- “road” feature could have a larger contribution to gradients
- The car can be too tiny or blurry that the model memorizes it by overfitting noise parts of the images

→ data augmentation could make bad & easy features harder to detect

Data augmentation as feature manipulation

Consider three types of features

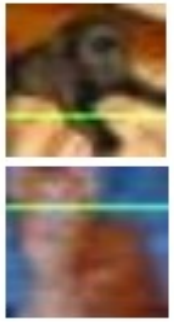
1. "good" & "easy to learn"
   – accurate features with large contribution in gradients
2. "good" & "hard to learn"
   – accurate features with small contribution to gradients
3. "bad" & "easy to learn"
   – inaccurate features with large contribution to gradients

Gradient descent learns by fitting data with (1)&(3) first before using (2)

Data augmentation can be viewed as manipulation of relative contribution of "good" and "bad" features in the gradients, *i.e.,* make (2) -> (1), or make (3) -> "bad" & "hard to learn"

Data augmentation as feature manipulation

Cat/Dog images with some spurious feature

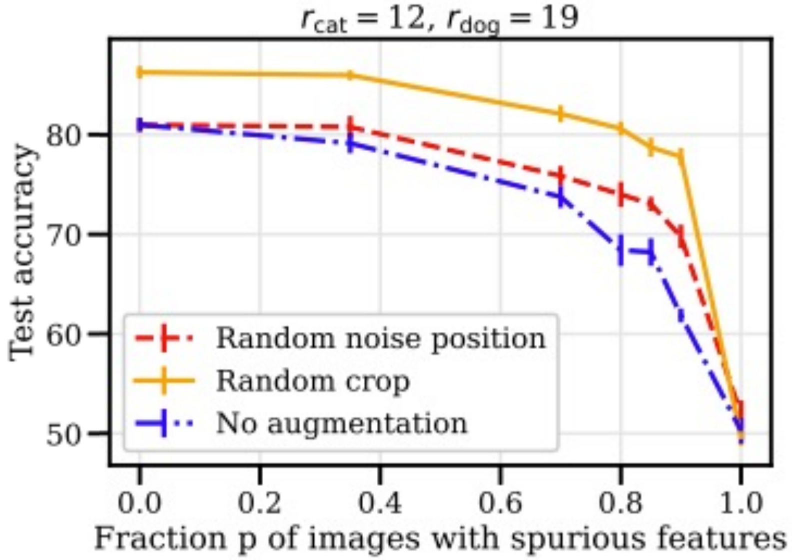


Baseline

Random noise position

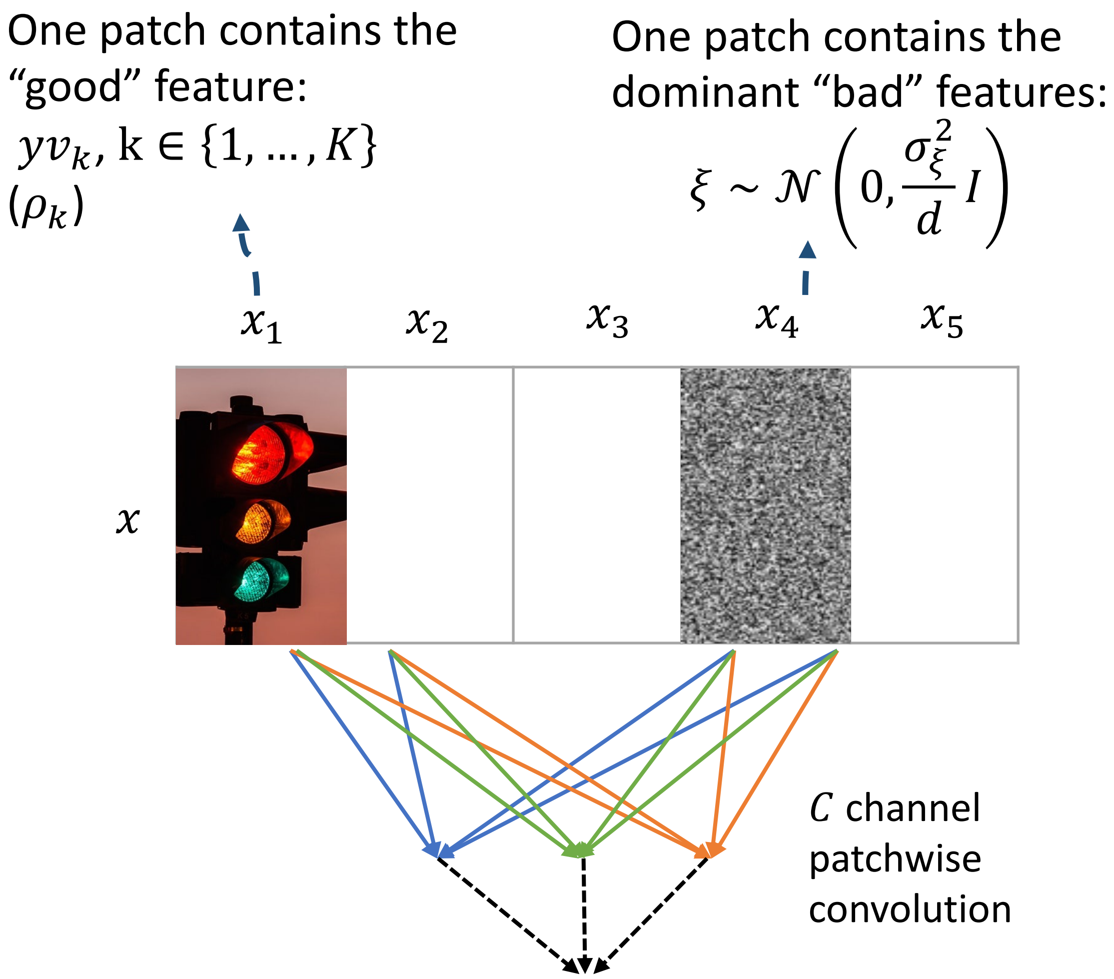
Random crop

- Random noise position: make noise harder to learn
- Random crop: make noise harder to learn + make good feature easier to learn

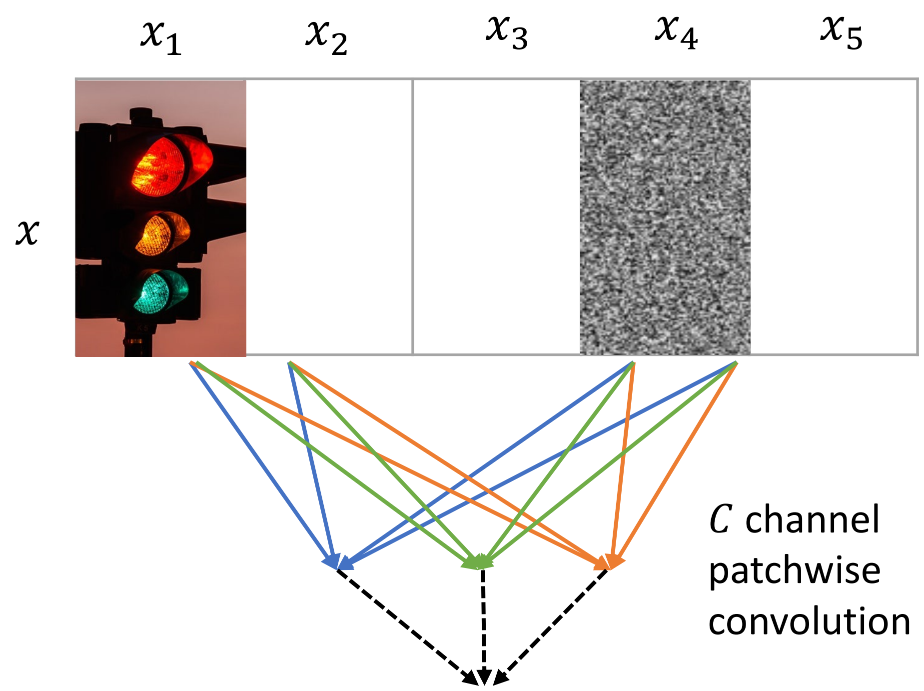


Learning & Generalization: Multi-view data model Allen-Zhu & Li (2019)

- Two classes $y \in \{-1,1\}$
- Inputs x has P patches $x = (x_1, x_2, \dots, x_P) \in \mathbb{R}^{d\times P}$
- Good features $v_1, v_2, \dots$
  - Data augmentation: $v_k \to v_{k'}$
  - A simplified model: One patch x_i contains feature v_k
- Noise feature ξ: One patch x_j contains ξ.



Patchwise convolutional model



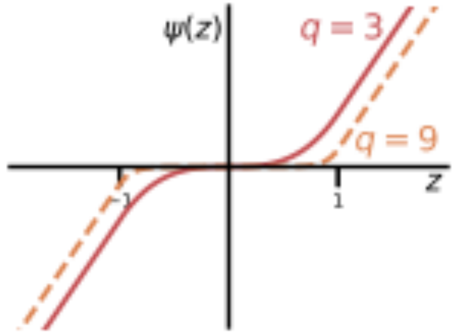
gradient descent on logistic loss

$$L(\mathbf{w}) = \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}} \text{ or } \mathcal{D}_{\text{train}}^{(\text{aug})}} \log(1+\exp(-yf(\mathbf{w},\mathbf{x})))$$

$$f(\mathbf{w},\mathbf{x}) = \sum_{c} \sum_{p} \psi(\mathbf{x}_p \cdot \mathbf{w}_c)$$

channels patches

$$\psi(z) = \begin{cases} \text{sign}(z) \cdot \frac{1}{q}|z|^q & \text{if } |z| \leq 1 \\ z - \frac{q-1}{q} & \text{if } z \geq 1 \\ z + \frac{q-1}{q} & \text{if } z \leq 1 \end{cases}$$



Learning dynamics with gradient descent

logistic loss $$L(\mathbf{w}) = \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}} \text{ or } \mathcal{D}_{\text{train}}^{(\text{aug})}} \log(1+\exp(-yf(\mathbf{w},\mathbf{x})))$$

over all datapoints over all patches

Learning dynamic of "good" feature v_k:

$$\frac{d}{dt} w_c \cdot v_k = \frac{1+o(1)}{2n} \sum_{i\in[n]} \sum_{p\in[P]} \psi'(|w_c \cdot x_p^{(i)}|) y^{(i)} x_p^{(i)} \cdot v_k$$

$$= \frac{1+o(1)}{2} \rho_k \psi'(|w_c \cdot v_k|)$$ + small order terms*

Fraction of datapoints with v_k

Learning dynamic of noise $\xi^{(i)}$:

$$\frac{d}{dt} w_c \cdot \xi^{(i)} = \frac{1+o(1)}{2n} \sum_{j\in[n]} \sum_{p\in[P]} \psi'(|w_c \cdot x_p^{(j)}|) y^{(j)} x_p^{(j)} \cdot \xi^{(i)}$$

$$= \frac{(1+o(1))\sigma_\xi^2}{2n} y^{(i)} \psi'(|w_c \cdot \xi^{(i)}|)$$ + small order terms*

*under assumptions on feature and noise

Learning dynamics with gradient descent

Learning dynamic of "good" feature v_k: $$\frac{d}{dt} w_c \cdot v_k \approx \rho_k \psi'(|w_c \cdot v_k|)$$

Learning dynamic of noise $\xi^{(i)}$: $$\frac{d}{dt} w_c \cdot \xi^{(i)} \approx \frac{1}{n} \sigma_\xi^2 y^{(i)} \psi'(|w_c \cdot \xi^{(i)}|)$$

$$f(\mathbf{w}, \mathbf{x}) = \sum_c \sum_p \psi(\mathbf{x}_p \cdot \mathbf{w}_c)$$

- For a datapoint with v_k and ξ, training accuracy is good if $w_c \cdot v_k$ large or $w_c \cdot y\xi$ large.
- If ρ_k is "small" compared to σ_ξ and n,
  - $w_c \cdot \xi$ grows faster than $w_c \cdot v_k$.
  - the model will classify the datapoint by overfitting to noise ξ.

Data augmentation:

- "good" and "hard" -> "good" and "easy": Increase ρ_k of rare views k.
- "bad" and "easy" -> "bad" and "hard": Increase n (through perturbing ξ).

Outline

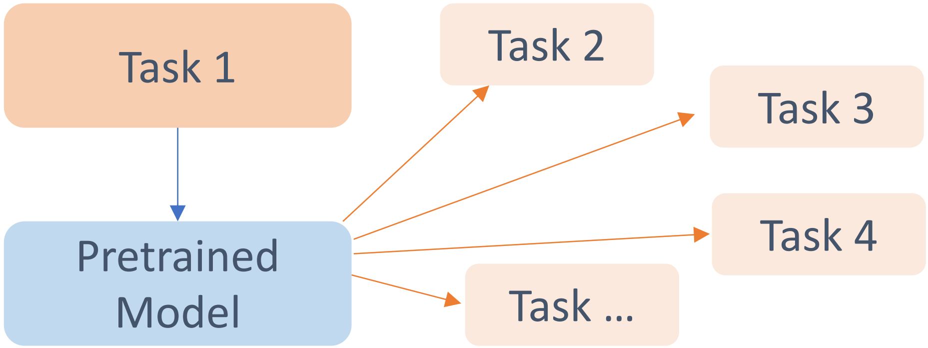
Understanding neural network training from the perspective of **feature learning**

- Motivation
- Two examples:
  - How does data augmentation help supervised learning tasks?
  - **How does using pretraining help classifying datasets with spurious correlations?**

Pretraining

Pretraining a model on a large dataset before transferring to a downstream can substantially improve accuracy over training from scratch.

e.g., ResNet-50 on unlabeled ImageNet boosts accuracy on CIFAR-10 from 94% to 98%



Task 1 — Large dataset

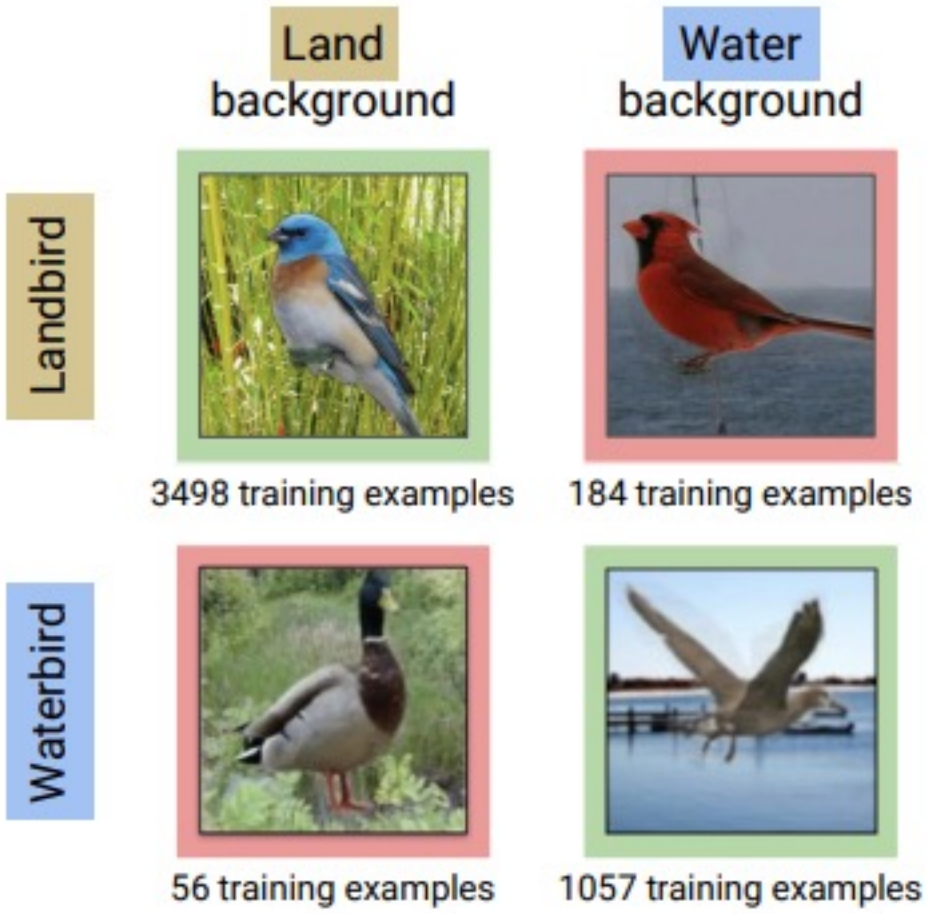
Vision: CLIP, DINO, …

NLP: Bert, Roberta, T5, GPT, ChatGPT,…

Task 2,3,4,… — Small dataset

Fine tuning/Freeze & Fine-tune/Linear Probing/…

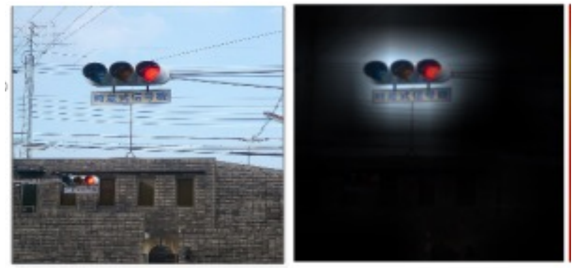
Waterbirds



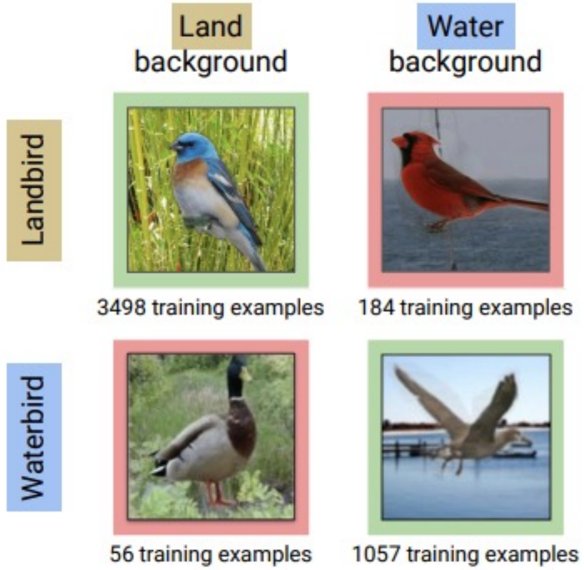
- Much more *waterbirds on water* (landbirds on land) than *waterbirds on land* (landbirds on water).
- In this dataset, the background feature is a spurious feature.
- SOTA results on Waterbirds (and other datasets with spurious correlations) uses pretrained model [Liu et al. 2021].
- Why does using pretrained model help?

Why using pretrained model help?

- Possibility 1:
  - Pretraining projects out the spurious feature (background feature).



Pretrained model learns to use the foreground to predict.

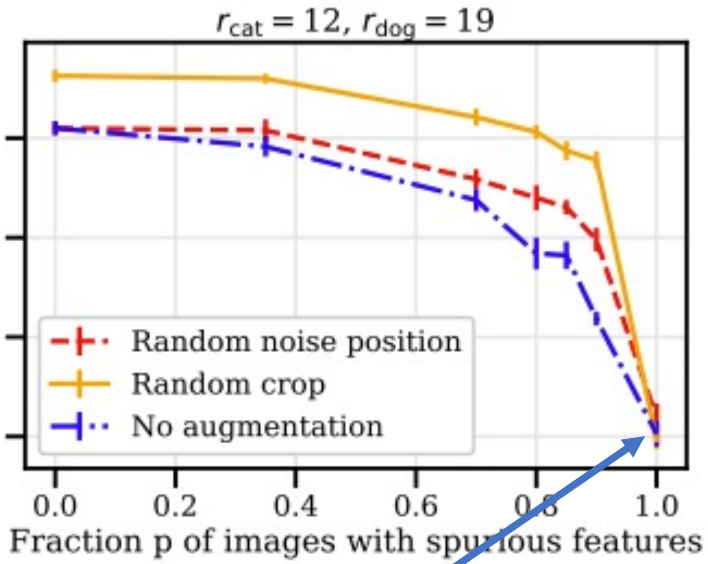


Use foreground(waterbird/landbird) to predict.

Why using pretrained model help?

- Possibility 1:
  - Pretraining projects out the spurious feature.

Cat/Dog images with spurious feature



Accuracy approaches 50% - random guess. The model does not learn any cat/dog feature at all.

Step 1: Pretrain ResNest20 on Cat/Dog without the spurious feature.
Step 2: Freeze & Fine-tune on Cat/Dog with 100% spurious feature.

Full fine-tuning	$52.87_{\pm 1.55}$
Freeze conv and block 1	$54.63_{\pm 1.07}$
Freeze conv and blocks 1-2	$68.37_{\pm 0.67}$
Freeze conv and blocks 1-3	$84.9_{\pm 0.46}$

Fig: Test accuracy on dataset without the spurious feature.

Why using pretrained model help?

- Possibility 1:
  - Pretraining projects out the spurious feature. ✓

Step 1: Pretrain ResNest20 on Cat/Dog without the spurious feature.
Step 2: Freeze & Fine-tune on Cat/Dog with the spurious feature.

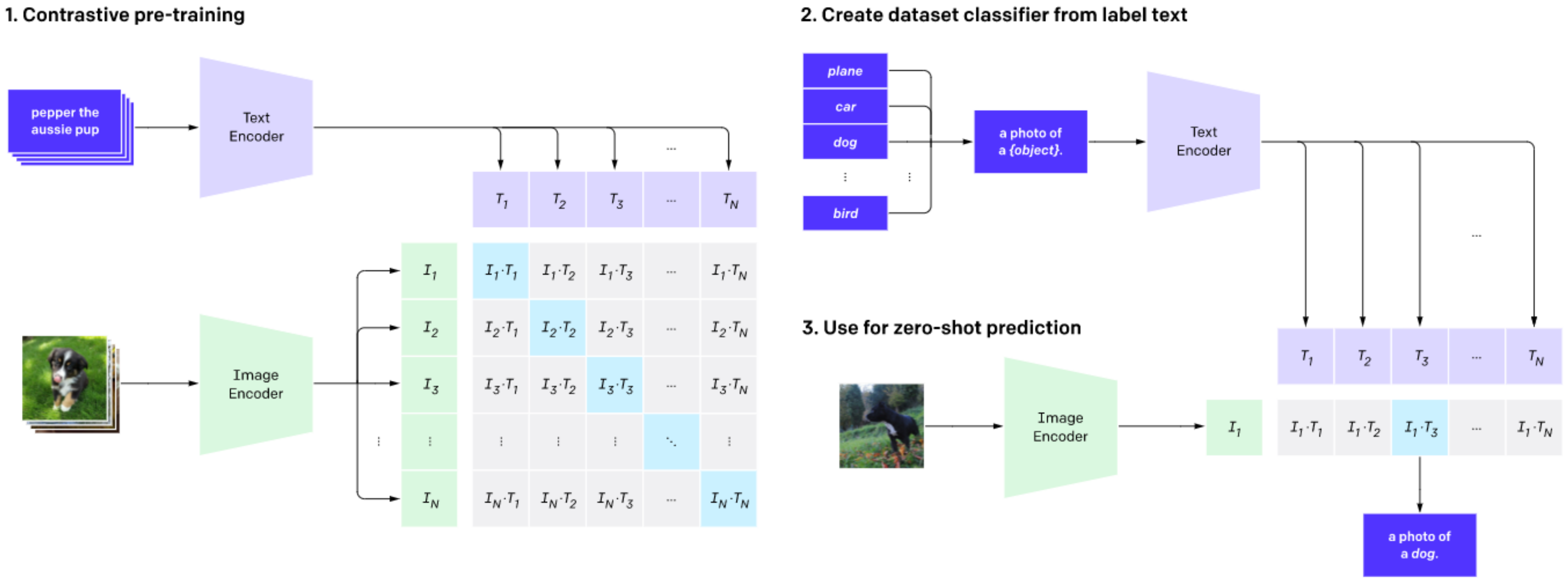
Pretraining projects out the spurious feature this case.

BUT in this case, we pretrained a **small** model on a **small** dataset and the spurious feature is unnatural.

What about larger models?

CLIP

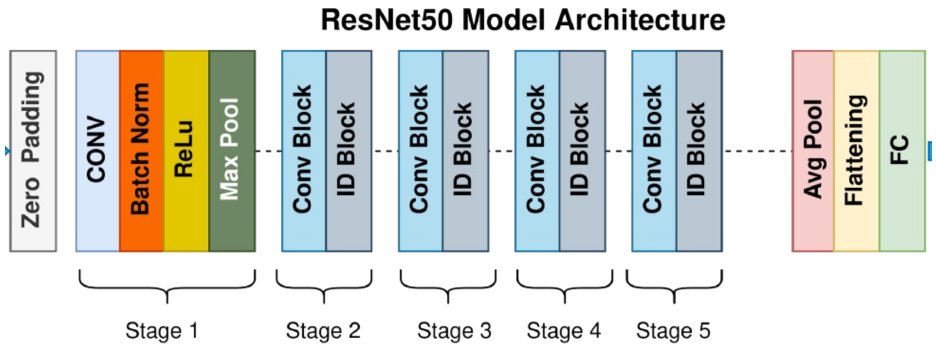
[Radford et al. 2021]

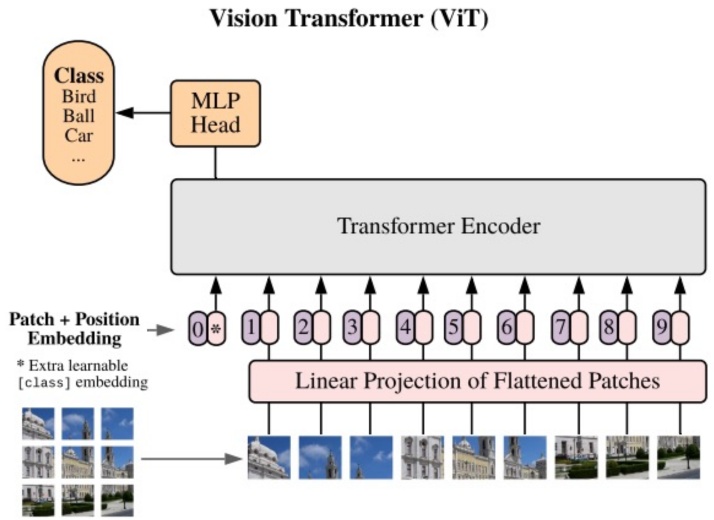


CLIP

- Radford et al. trained CLIP on 5 ResNets and 3 Vision Transformers.
  - ResNet-50, ResNet-101, RN50x4, RN50x16, and RN50x64
  - ViT-B/32, a ViT-B/16, and a ViT-L/14

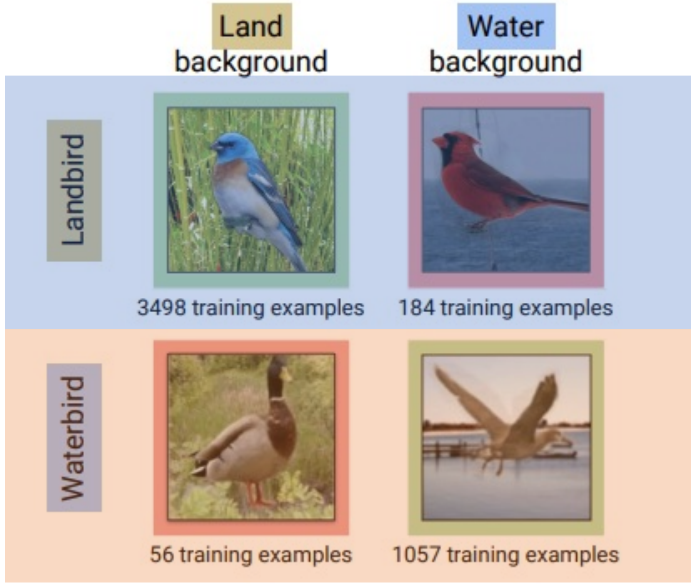
- Pretrained on a WebImageText (WIT) dataset
  - 37.6 million entity rich image-text examples with 11.5 million unique images across 108 Wikipedia languages



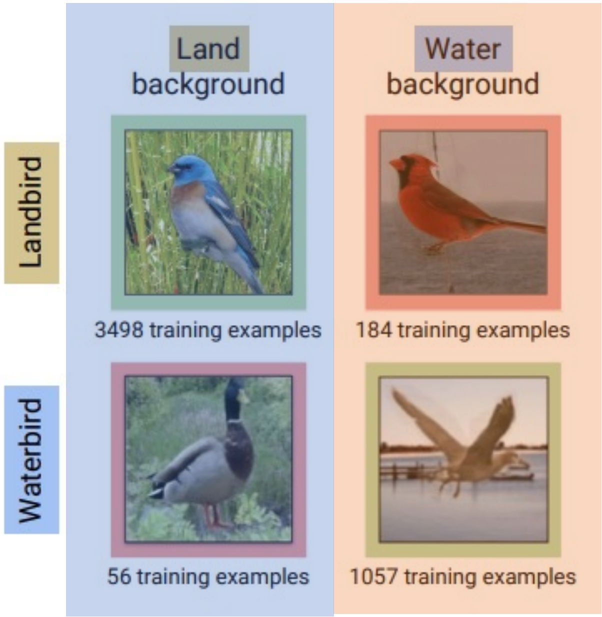


Why using pretrained model help?

- Possibility 1:
  - Pretraining projects out the spurious feature (background feature).
- How do we test? We consider two tasks on the waterbirds dataset.



Foreground Prediction



Background Prediction

Why using pretrained model help?

- Possibility 1:
  - Pretraining projects out the spurious feature (background feature).

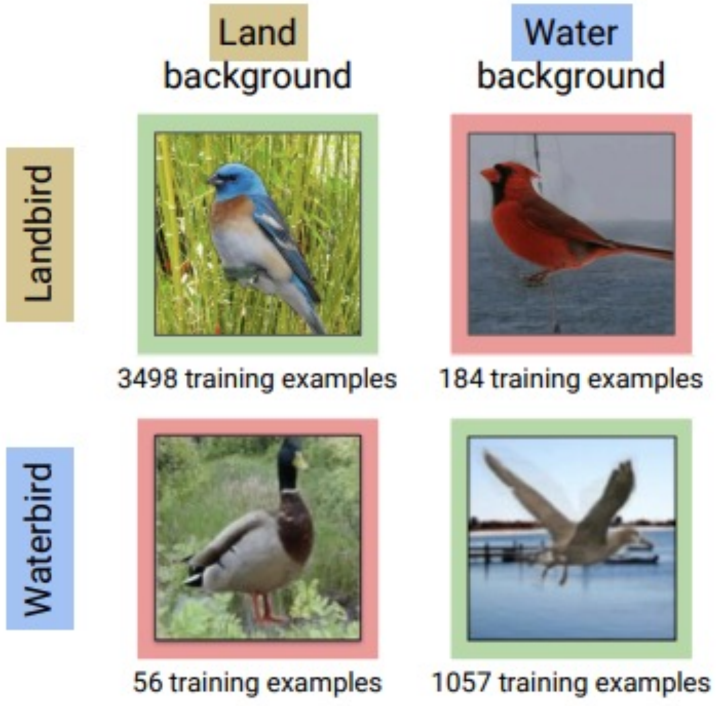
	Foreground	Background
Full Fine-tuning	61.99	88.96
Freeze embed	74.04	88.78
Freeze embed & layers 1-3	73.94	87.85
Freeze embed & layers 1-6	**76.01**	**89.15**
Freeze embed & layers 1-9	72.79	89.12
Freeze embed & layers 1-12	74.66	88.94

Fig: Worst group accuracy of fine-tuning CLIP ViT-B/16 on Waterbirds.

- The amount of information preserved from pretraining:
  Full Fine-tuning < Freeze embed < freeze embed & layer 1-3 < …
- Accuracy increases as we preserving more information.
  (Freezing too many layer is bad because there won't be enough capacity to adapt to the downstream task)
- Preserving information from the pretrained model helps both foreground prediction and background prediction.
  -> **Pretraining does not project out the background.**

Why using pretrained model help?

- Possibility 2:
  - Pretraining projects out the noise.



How does the model overfit?
If the true feature is not used,
- Overfit the spurious feature (background)

Classify Waterbirds on Water and Landbirds on Land correctly.
What about Waterbirds on Land and Landbirds on Water?
- Overfit the noise

Preventing the model from overfitting the noise can also motivate the model to use the true feature!

Theory

- Inputs x has P patches $x = (x_1, x_2, \dots, x_P) \in \mathbb{R}^{d \times P}$
- Good features v
- Noise feature ξ
- Spurious features u

How fast the model learns a feature depends on the ***magnitude*** and the ***frequency*** of the feature.

Learning dynamic of feature v:

$$\frac{d}{dt} w_c \cdot v = \frac{1 + o(1)}{2} \rho \psi'(|w_c \cdot v|) \|v\|_2^2$$ + small order terms*

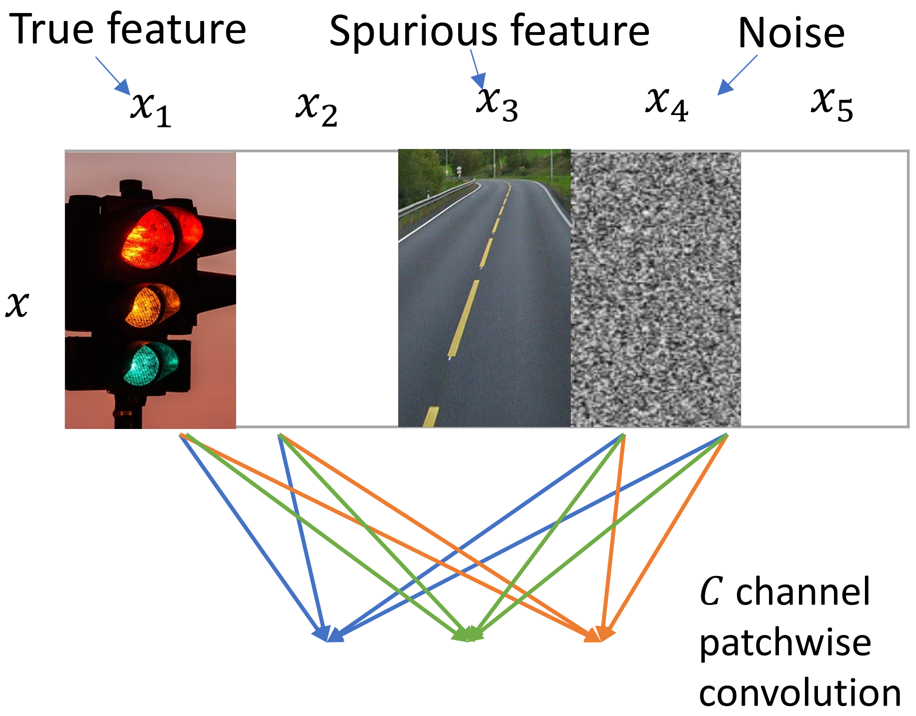
Frequency

Magnitude

Pretraining diminishes the magnitude of noise/spurious feature

-> The model can't use the noise/spurious feature to overfit

-> The model is forced to use the true feature



Summary

- Understanding neural network training from the perspective of feature learning
  - Both theoretically and empirically
  - Accurate & insightful perspective

Thank you!