Lecture 24: Midterm 2 review

CS 182 (“Deep Learning”)

2022/04/25

Today’s lecture

- Today’s lecture is for 182 students only, and it is the last lecture
- No lecture Wednesday! You have midterm 2 instead
- Same high level points as the last midterm review lecture: no new content, not a substitute for studying on your own, we may not get through all slides, etc.
- Reviewing the last midterm review slide deck is also a good idea

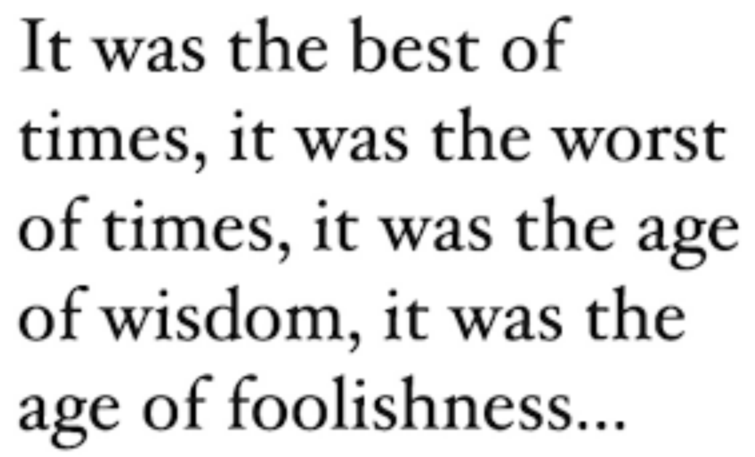
Midterm 2 logistics

- For all students with standard accommodations (if you’re not sure, this is you):
  - Midterm time is 5-7pm — arrive promptly at 5pm, we begin promptly at 5:10
  - Everyone is here in this room (Dwinelle 155)
- Students with DSP accommodations: make a private Piazza post if you have not yet received your specific logistics
- One double sided 8.5x11in cheat sheet is permitted

Modeling sequential data

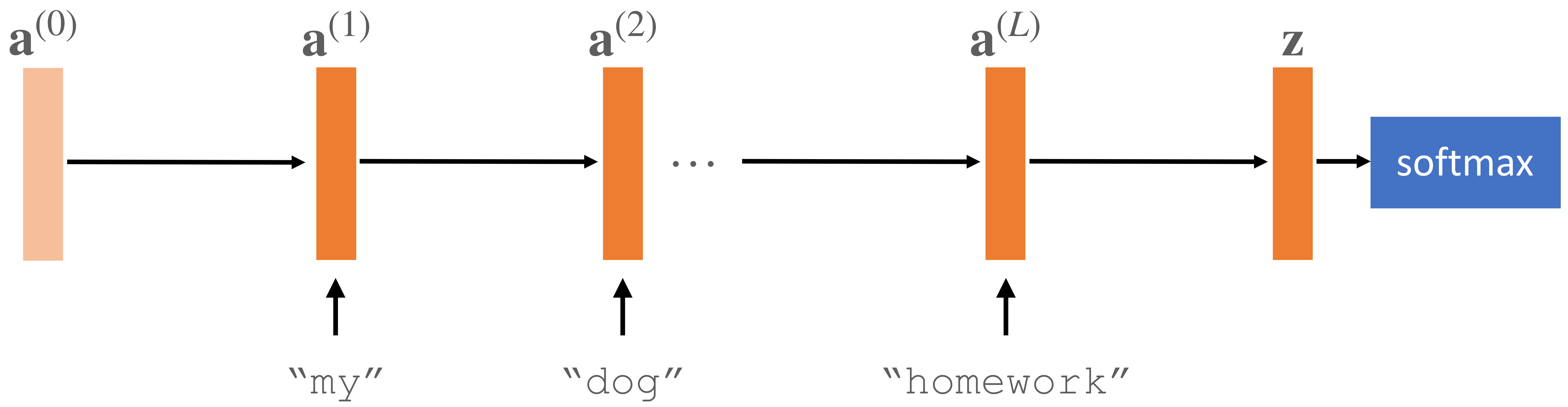
Problem setup

- We now consider settings in which our features $\mathbf{x}$ represent *sequential data* which may be *variable length*


It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness...

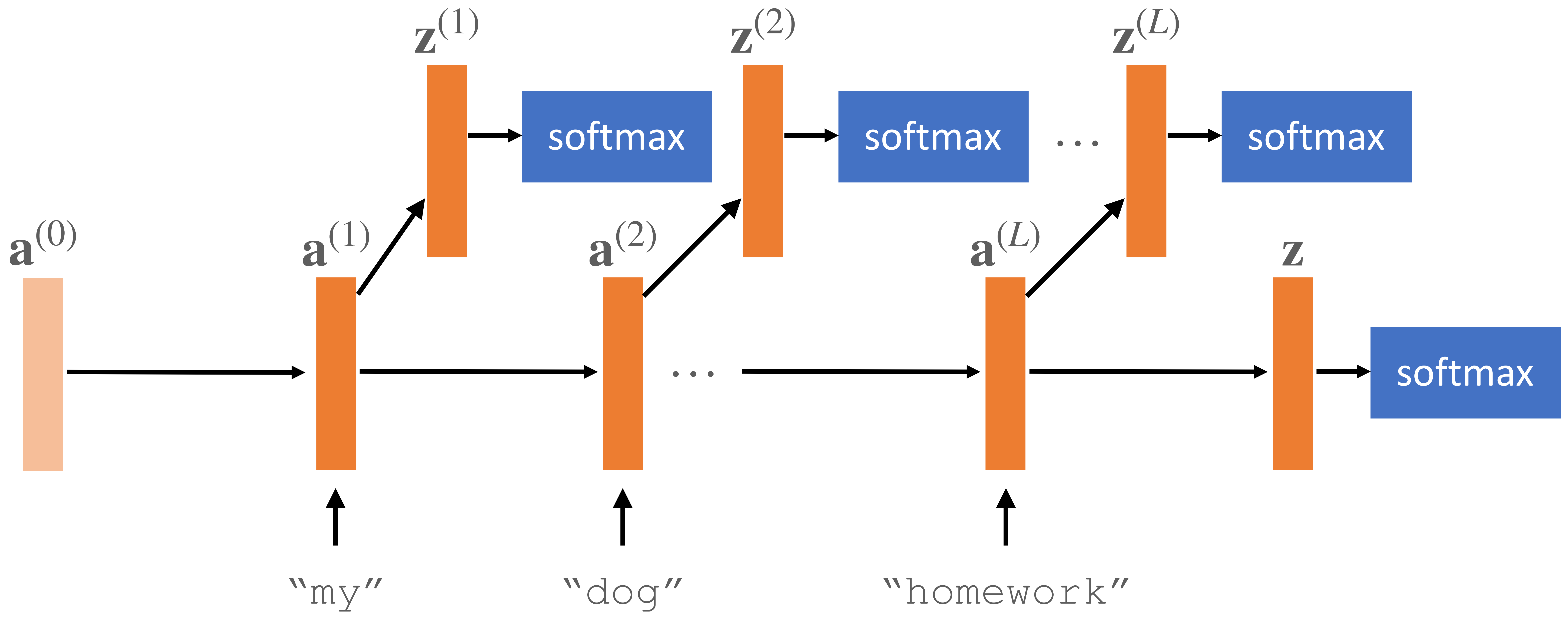
- Our labels could be scalars y, e.g., sentiment analysis, identification, …
- Or the labels could be sequences $\mathbf{y}$! E.g., translation, transcription, captioning, …
- Or there could be no label at all! I.e., **unsupervised learning / generative modeling**

Recurrent networks



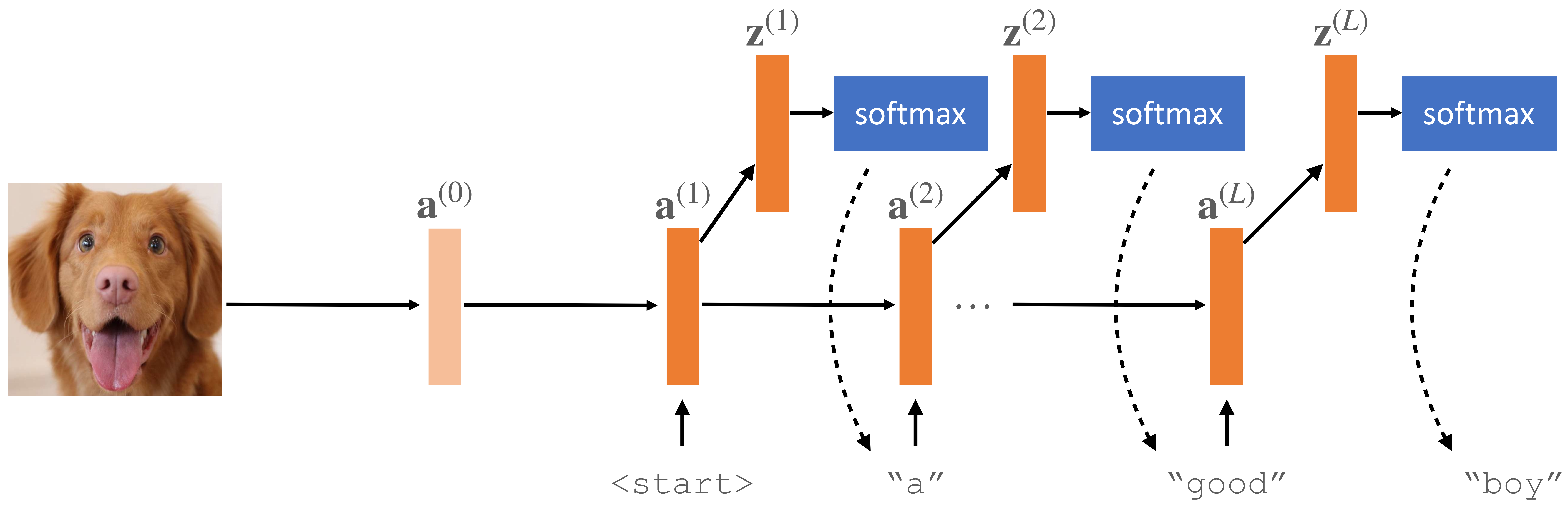
- What does this look like, mathematically, both for "vanilla" RNNs and LSTMs?
- In many applications, we think of each l as a "time step" (denoted t instead) and each $\mathbf{a}^{(l)}$ as the "state" (or *hidden state*) at time step l (denoted $\mathbf{h}^{(t)}$ instead)

Sequential outputs



- This is what our RNN will look like for "sequence input, single output"
  - What about sequence output? Just have an output at every layer

Autoregressive generation



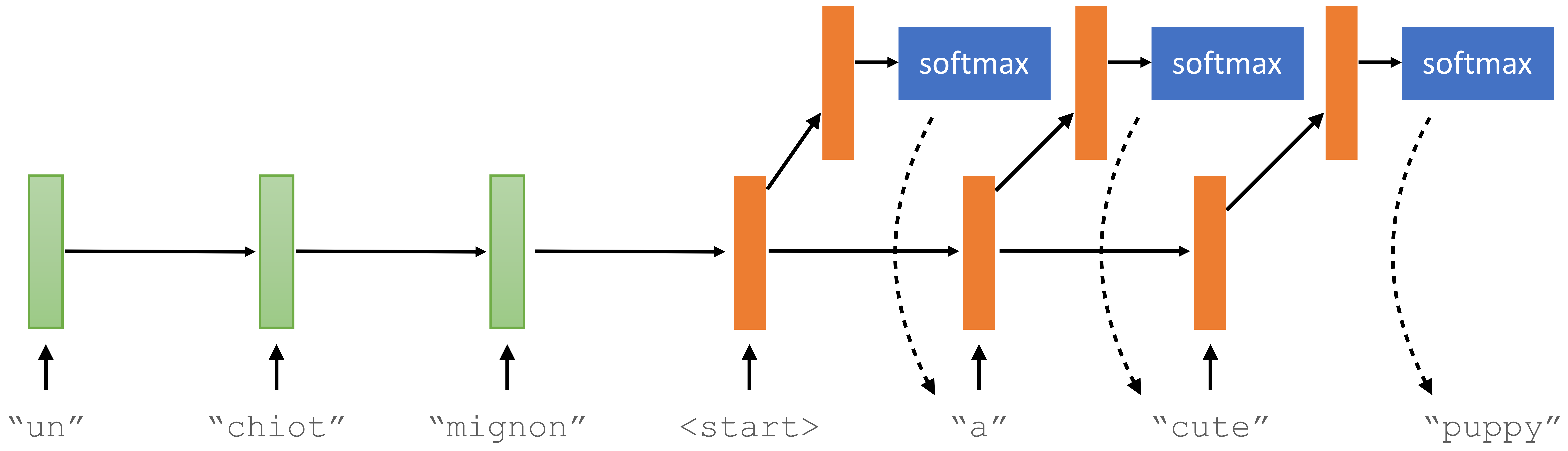
- Generating a sequential output from an RNN, e.g., to caption an input image, is done in an **autoregressive** manner
    - This makes it possible for the RNN to condition on what it has already generated

Transformers

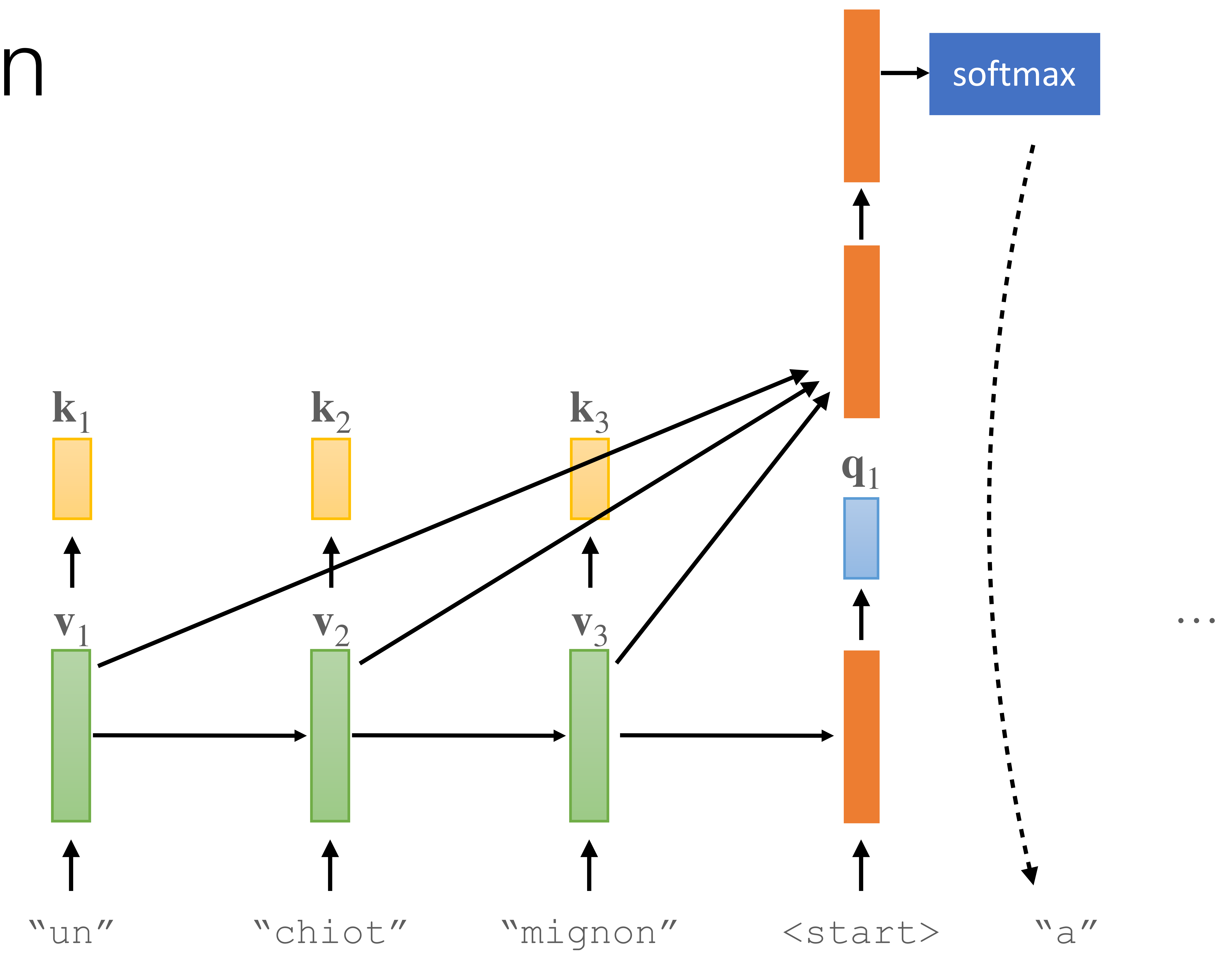
- Be familiar with the details of how attention and self-attention work
- Understand the differences between transformer encoders and decoders
- Be familiar with all of the transformer details we went over
- Understand all of the transformer based models we went over, in particular: BERT, GPT, ViT, MAE
    - How are they trained, how are they evaluated, what are they good/bad at, etc.

Sequence to sequence models

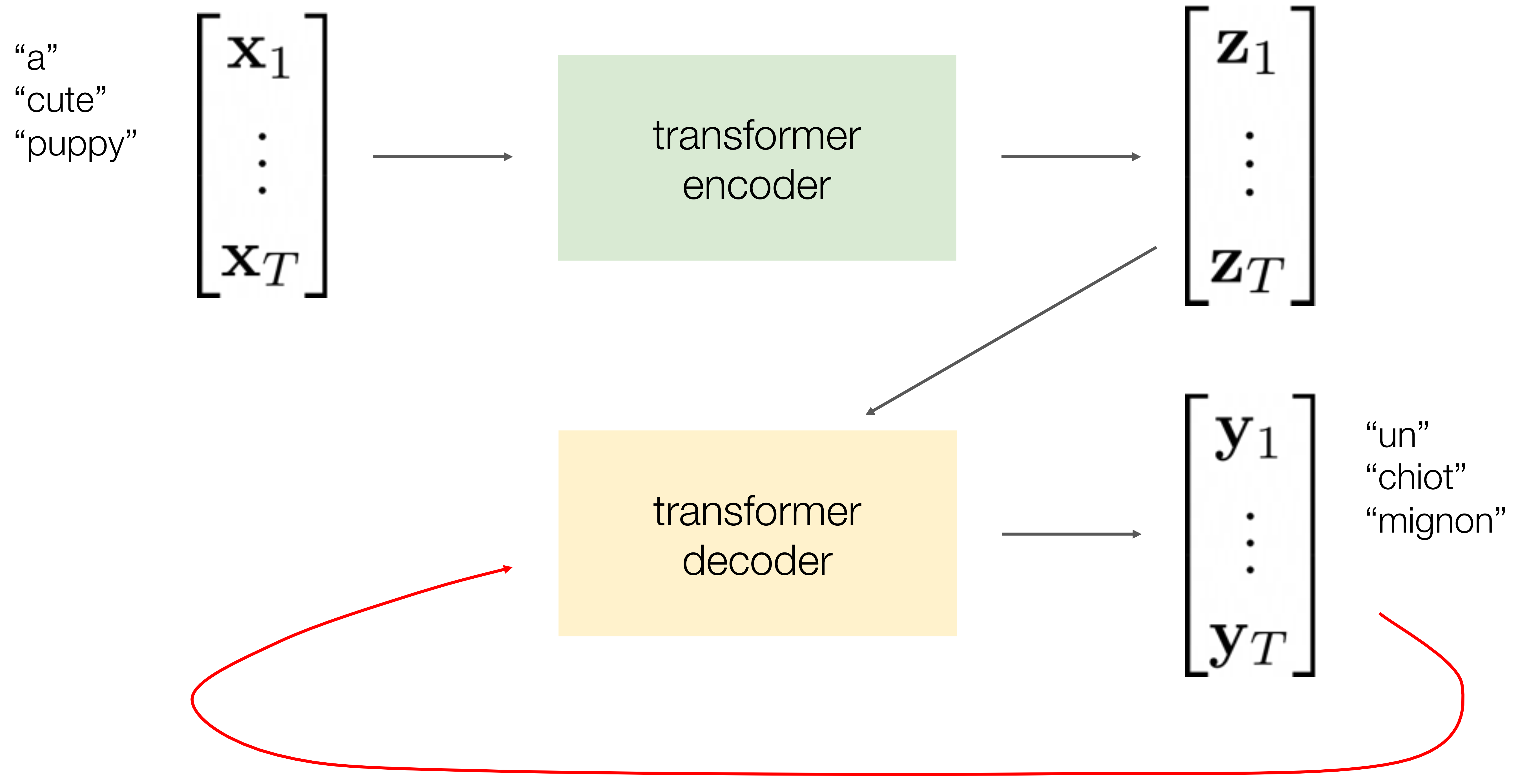
RNN (LSTM) seq2seq, the basic version



Attention


softmax
$\mathbf{k}_1$
$\mathbf{k}_2$
$\mathbf{k}_3$
$\mathbf{q}_1$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
…
"un"
"chiot"
"mignon"
<start>
"a"

Seq2seq transformers



"a"
"cute"
"puppy"
$\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}$
transformer encoder
$\begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_T \end{bmatrix}$
transformer decoder
$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_T \end{bmatrix}$
"un"
"chiot"
"mignon"

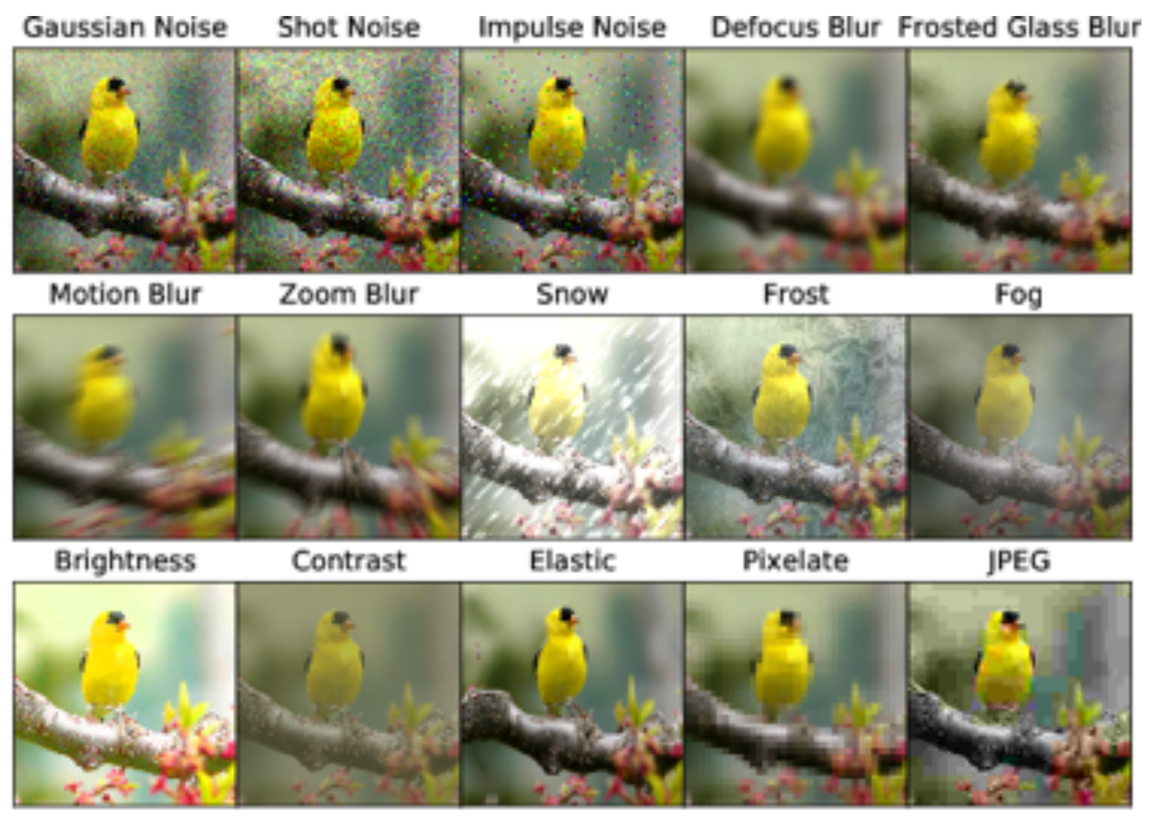
Natural language processing guest lecture

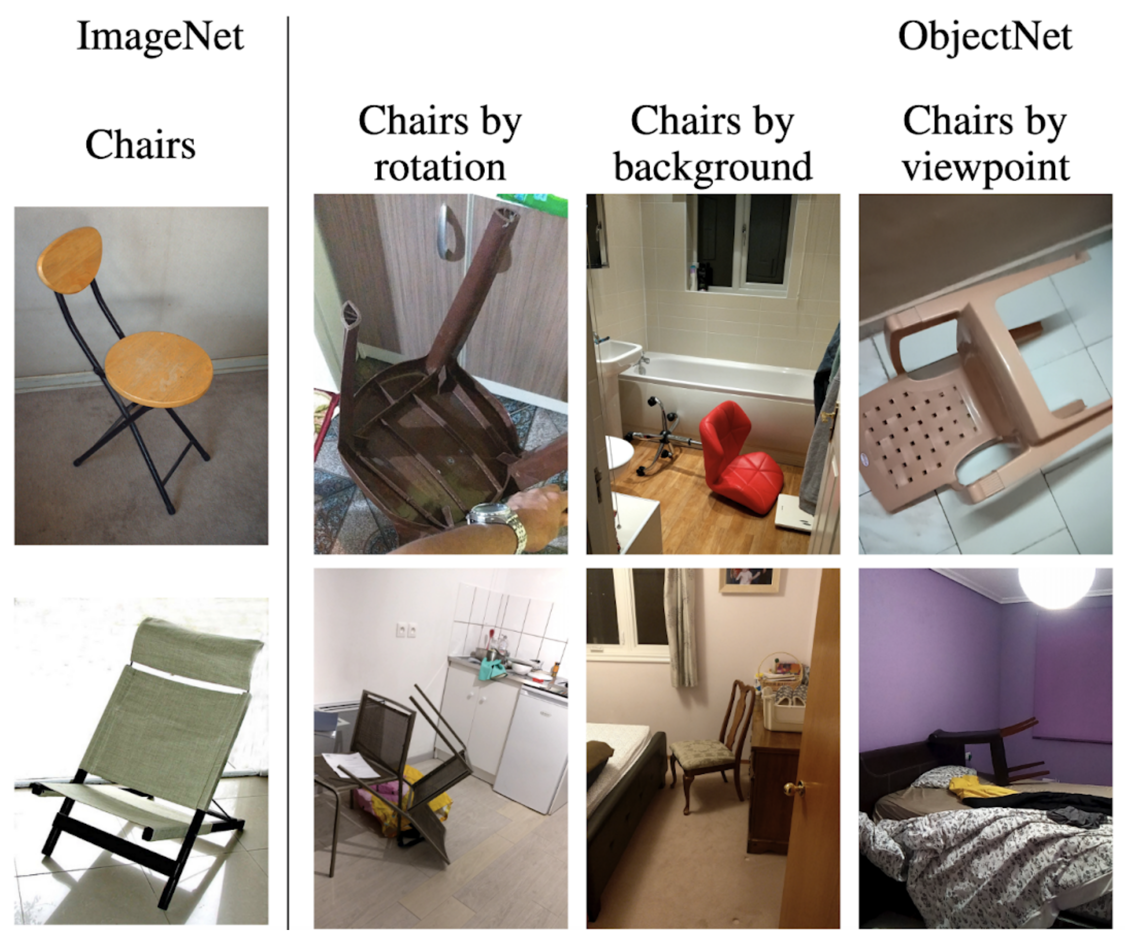
- Understand the details behind decoding: beam search and item scoring
- Understand training details such as teacher forcing, label smoothing
- Understand how text is actually represented as tokens, specifically, BPE
- Everything up to slide 23 (first slide about multilingual translation) is fair game, this is how far John got in lecture

Distribution shift and robustness

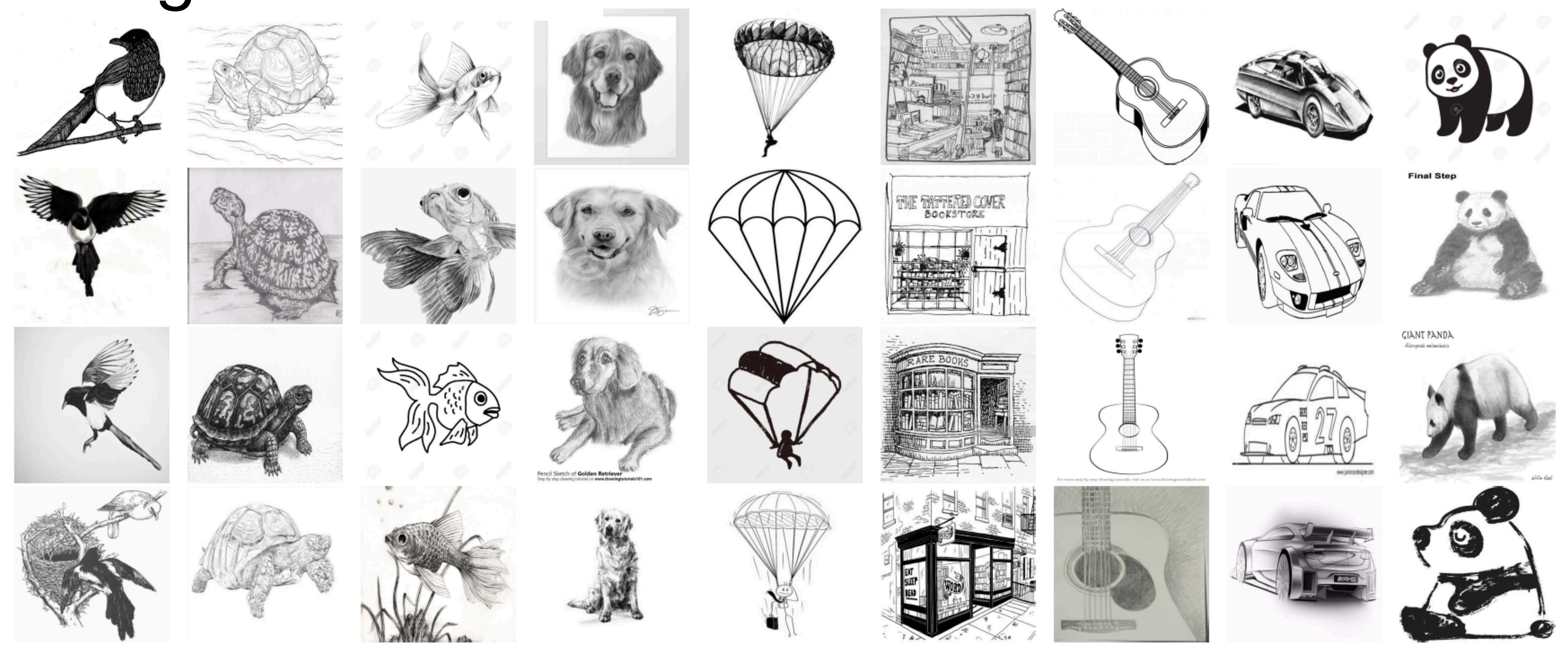
ImageNet challenge test sets

ImageNet-C





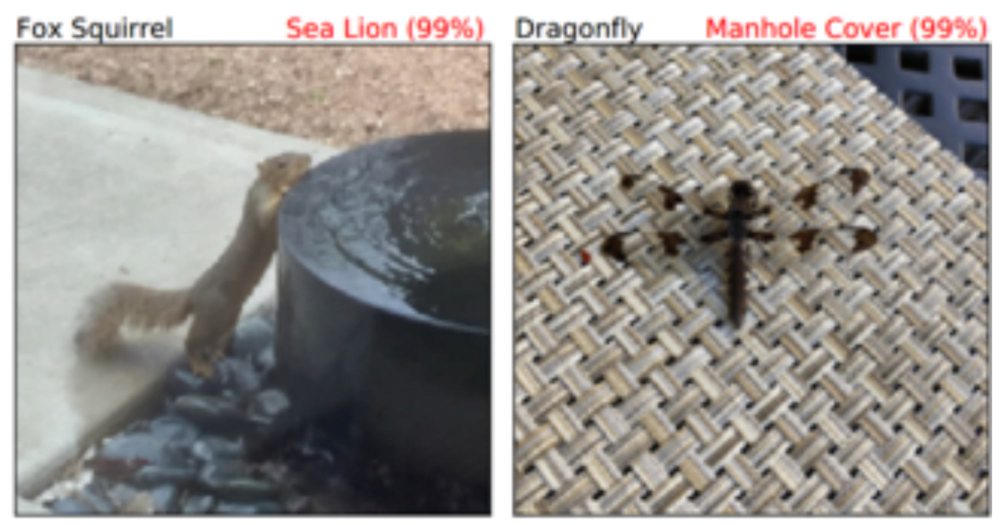
ImageNet-Sketch



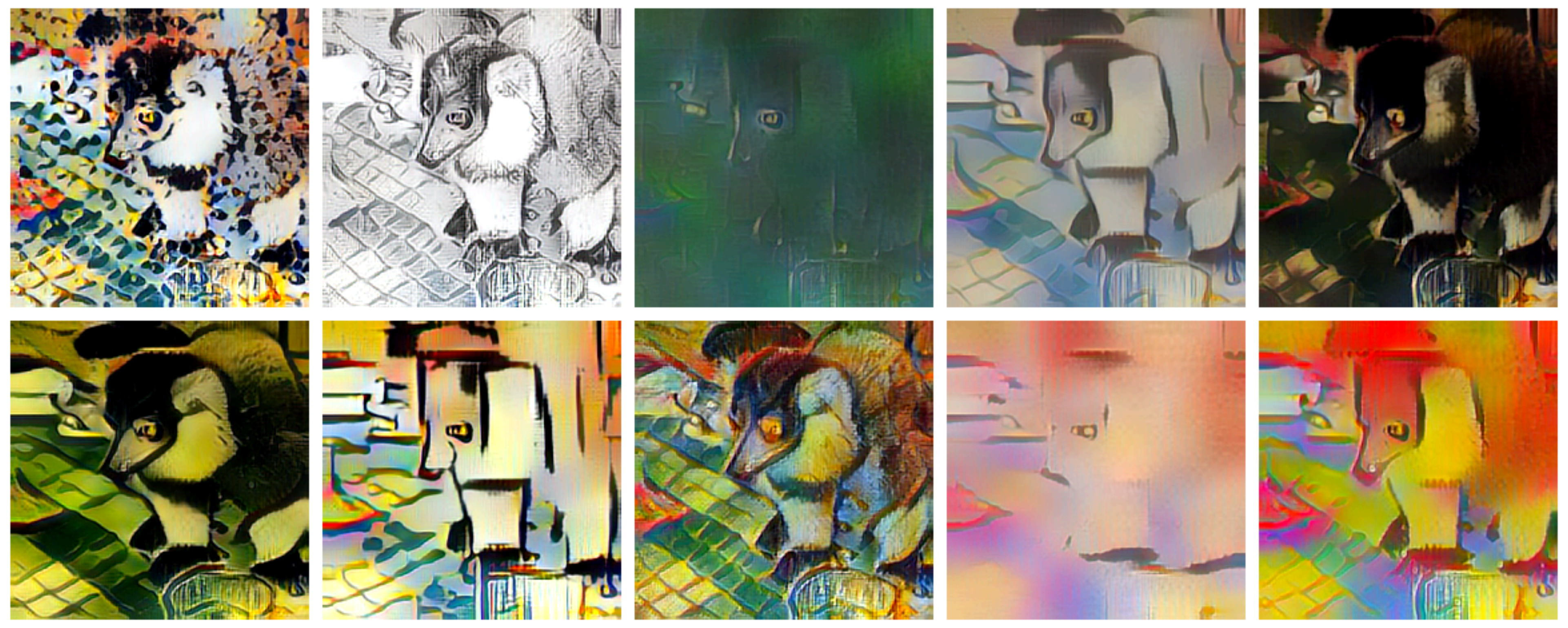
ImageNet-R

Painting
Origami

ImageNet-A

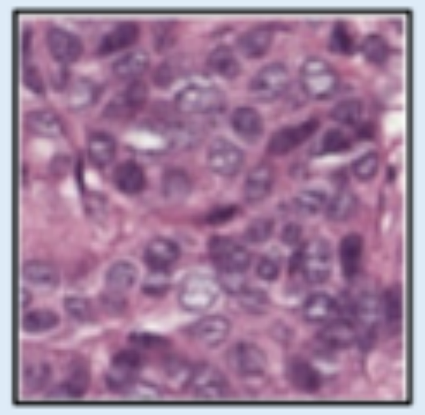
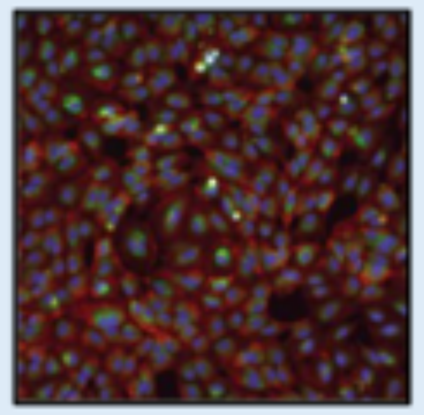
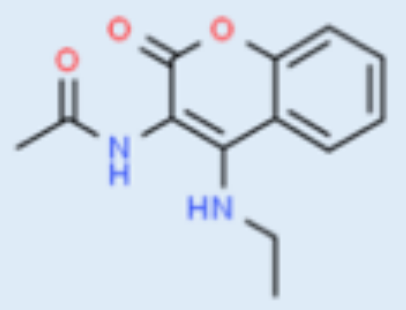
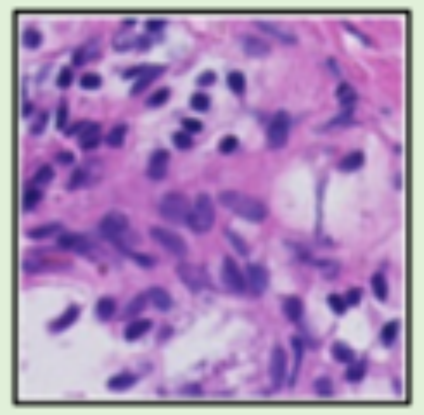
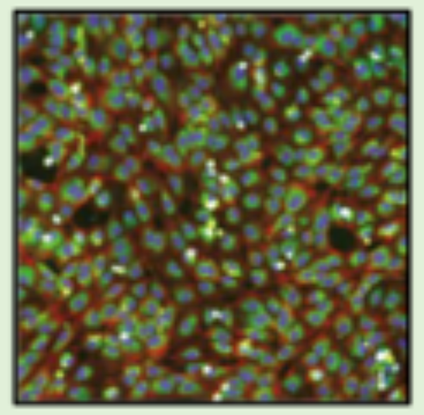
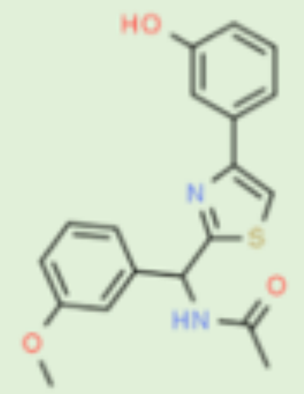
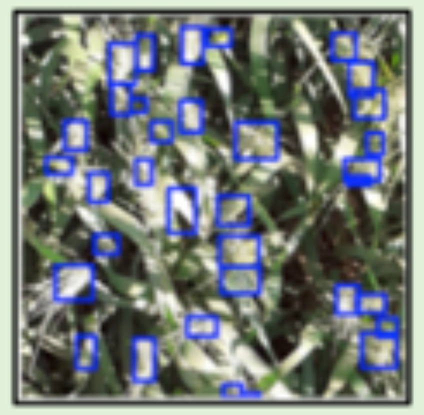


Stylized ImageNet



The WILDS benchmark

https://wilds.stanford.edu

Dataset	iWildCam	Camelyon17	RxRx1	OGB-MolPCBA	GlobalWheat	CivilComments	FMoW	PovertyMap	Amazon	Py150
Train example						What do Black and LGBT people have to do with bicycle licensing?			Overall a solid package that has a good quality of construction for the price.	`import numpy as np … norm=np.___`
Test example						As a Christian, I will not be patronizing any of those businesses.			I *loved* my French press, it's so perfect and came with all this fun stuff!	`import subprocess as sp p=sp.Popen() stdout=p.___`
Adapted from	Beery et al. 2020	Bandi et al. 2018	Taylor et al. 2019	Hu et al. 2020	David et al. 2021	Borkan et al. 2019	Christie et al. 2018	Yeh et al. 2020	Ni et al. 2019	Raychev et al. 2016
Domain (d)	camera	hospital	batch	scaffold	location, time	demographic	time, region	country, rural-urban	user	git repository

In NLP: the ANLI dataset

- *Natural language inference* is the task of determining if a premise sentence and hypothesis sentence are related through contradiction, neutrality, or entailment
- The **adversarial natural language inference (ANLI) dataset** consists of crowdsourced hypotheses written to fool state-of-the-art models
- To construct the dataset: an annotator is asked to write a hypothesis given a premise and a condition (contradiction, neutrality, or entailment)
  - If the model correctly predicts the condition, the annotator is asked to try again
  - If the model predicts incorrectly, the hypothesis is verified by other annotators

What improves distributional robustness?

- Training larger models on larger datasets
- Strong data augmentations, e.g., Mixup, AutoAugment, AugMix, PixMix
- Better architectures and training objectives: the current state of the art numbers for ImageNet-C, R, A, and Sketch (using only the ImageNet training set) are obtained with ViT models pretrained with a masked autoencoding objective
- Be familiar with the details of all of the above

Anomaly detection: the basics

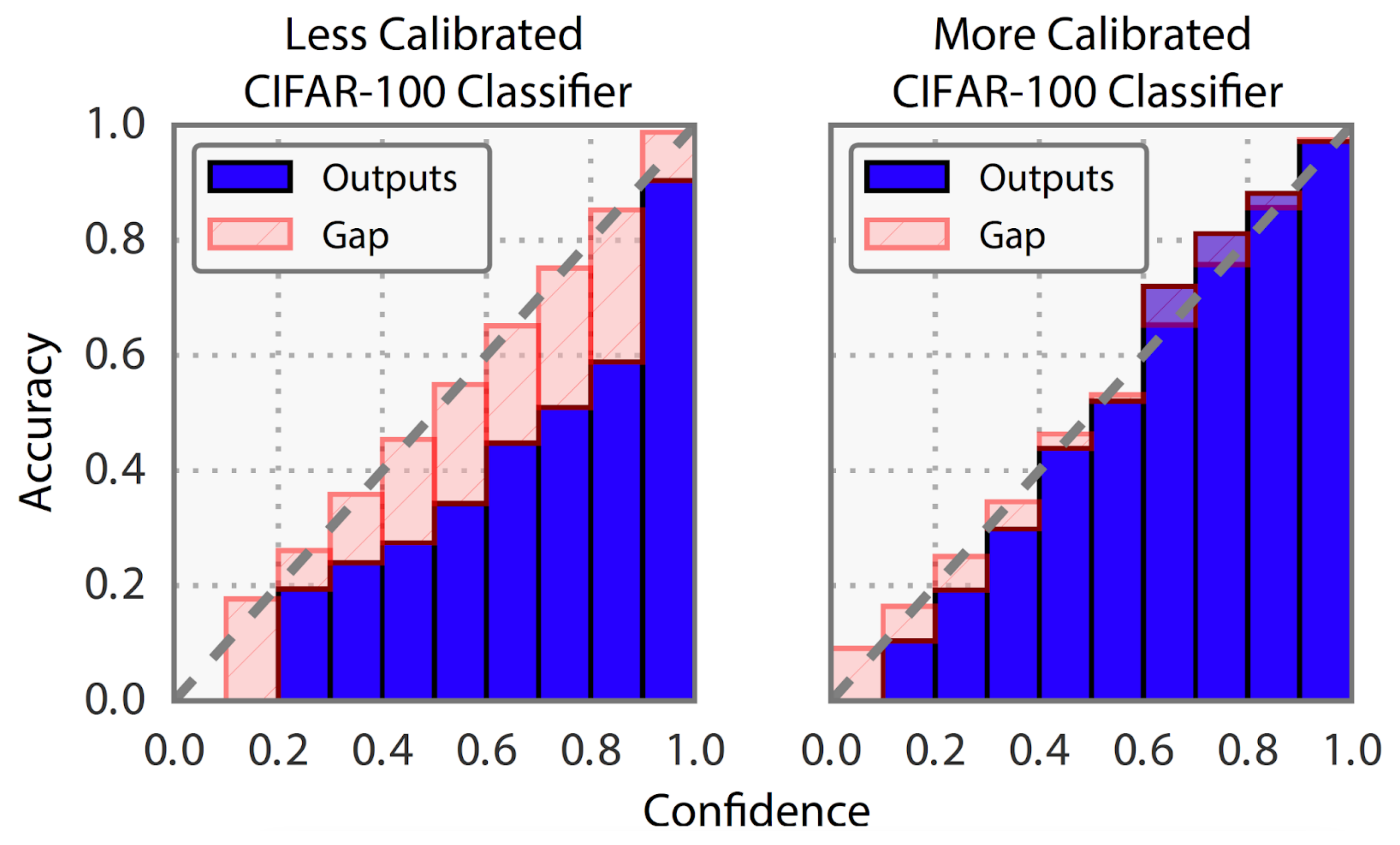
- We would like for our model to assign an **anomaly score** to every input $\mathbf{x}$ — the higher the score, the more anomalous the model thinks the example is
- An intuitive idea would be to try and learn a model of $p(\mathbf{x})$ (a *generative model*) and treat an $\mathbf{x}$ as anomalous if it has low $p(\mathbf{x})$ according to the model
    - This currently does not work well! Modern deep generative models often still do poorly at anomaly detection using this scheme for complex input spaces
- There are some ways to make deep generative models useful for anomaly detection, though they are more complex and require additional assumptions

A simple baseline for anomaly detection

- A better approach that does not involve training a generative model is to use the model's **confidence** $\max_k p_\theta(y = k \mid \mathbf{x})$ to detect anomalies
  - Specifically, use $-\max_k p_\theta(y = k \mid \mathbf{x})$ as the anomaly score
  - In some contexts, $-\max_k \mathbf{z}_k$ (negative of max logit) may work better
- This simple baseline works reliably across computer vision, NLP, and speech recognition classification tasks, though it can't detect *adversarial examples*

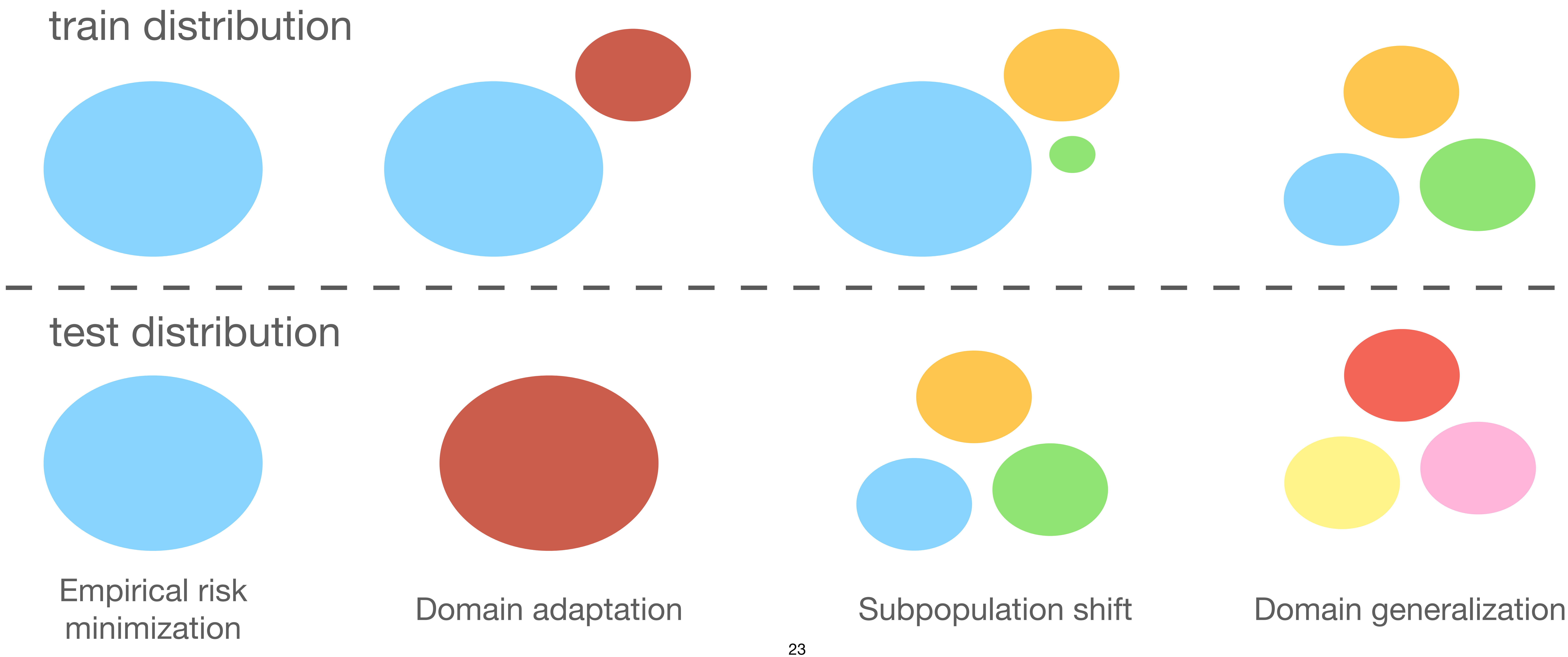
Model calibration

- Another concept related to the general reliability of machine learning models, but not tied to distribution shift, is model **calibration**
- We measure calibration by comparing a model's confidence against its accuracy
- Well calibrated models are more trustworthy, easier to integrate, and more interpretable
- Calibration under distribution shift is hard
- Understand the basics behind temperature scaling and deep neural network ensembles



Characterizing real-world distribution shifts

Problem settings and frameworks



Domain adaptation

- What if we knew at training time which test distribution we want to do well on?
- In this case, shouldn't we just train with the test distribution?
    - The issue is that data from the test distribution may be difficult to collect
- So, the assumption (as typically stated) made by **domain adaptation** is that we have abundant training data from a **source domain** and only a small amount of training data from the **target domain** which is the actual domain of interest
- E.g., source vs. target could be simulation vs. the real world
- Or, e.g., the overall population vs. an underrepresented group of interest

Invariant feature learning

- At a high level, invariance in this context (usually) means that we wish for the feature distributions between the source and target data to look identical
- Intuitively, if the model is outputting similar features for both source and target data and predicting well for source data, we may expect that it will also predict well on target data
- A few approaches have been proposed for invariant feature learning: trying to fool learned *domain discriminators* and matching distribution statistics between the source and target data
    - Be sure to understand these approaches at a high level

Subpopulation shift

- In **subpopulation shift**, we assume several training domains rather than just two
    - Domains are also referred to as “groups” or “subpopulations” in this context
- The key challenge in subpopulation shift is that some domains are *underrepresented* in the training data
- However, those domains may contribute significantly to the model’s generalization performance, either because they will be equally represented in the test distribution, or because we care about **fairness** across domains
    - In the latter case, it is natural to measure weighted or *worst-case* performance

Distributional (group) robustness

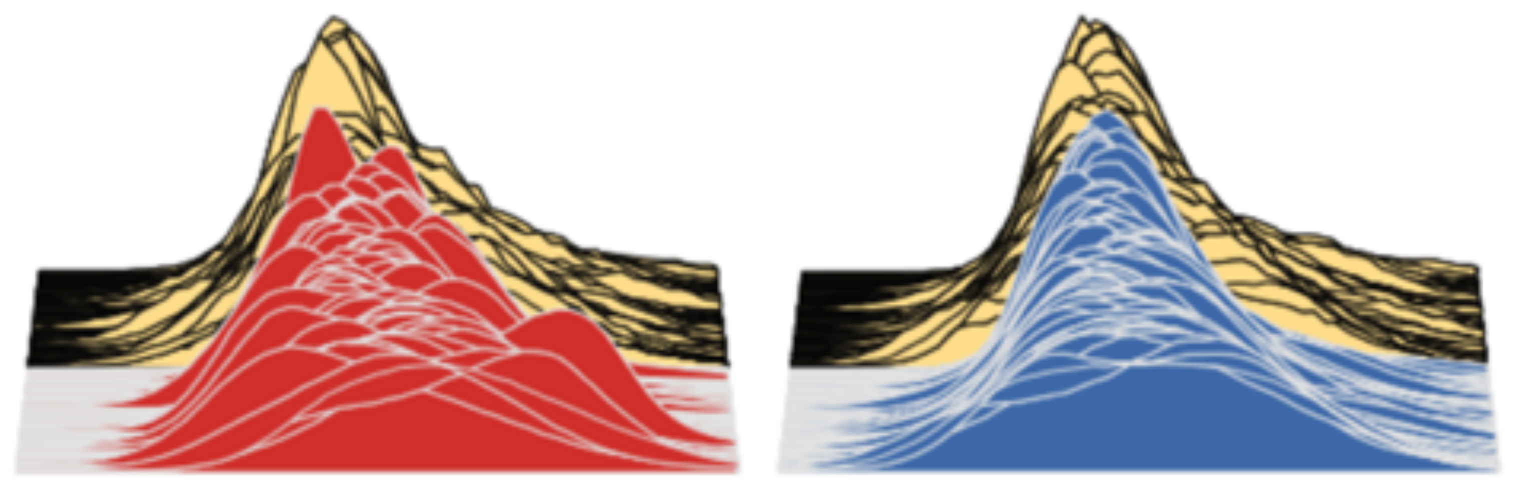
- Distributional robustness, in general, aims to train a model against an *adversary* that can change the data distribution to try and make the model worse
- In **group robustness**, the adversary is only allowed to change the distribution of domains
- Letting p_i be the probability of domain i, we have: $\min_{\theta} \max_{p_1,\ldots,p_D} \sum_{d=1}^{D} p_d \mathbb{E}_d[\ell(\theta; X, Y)]$
- **Rebalancing** the training data (by upsampling rare domains or downsampling common domains) turns out to be very effective at improving the worst-case performance
    - This is also a common and effective trick for handling class imbalance
    - We can sometimes further improve performance by weighting the loss function as well

Domain generalization

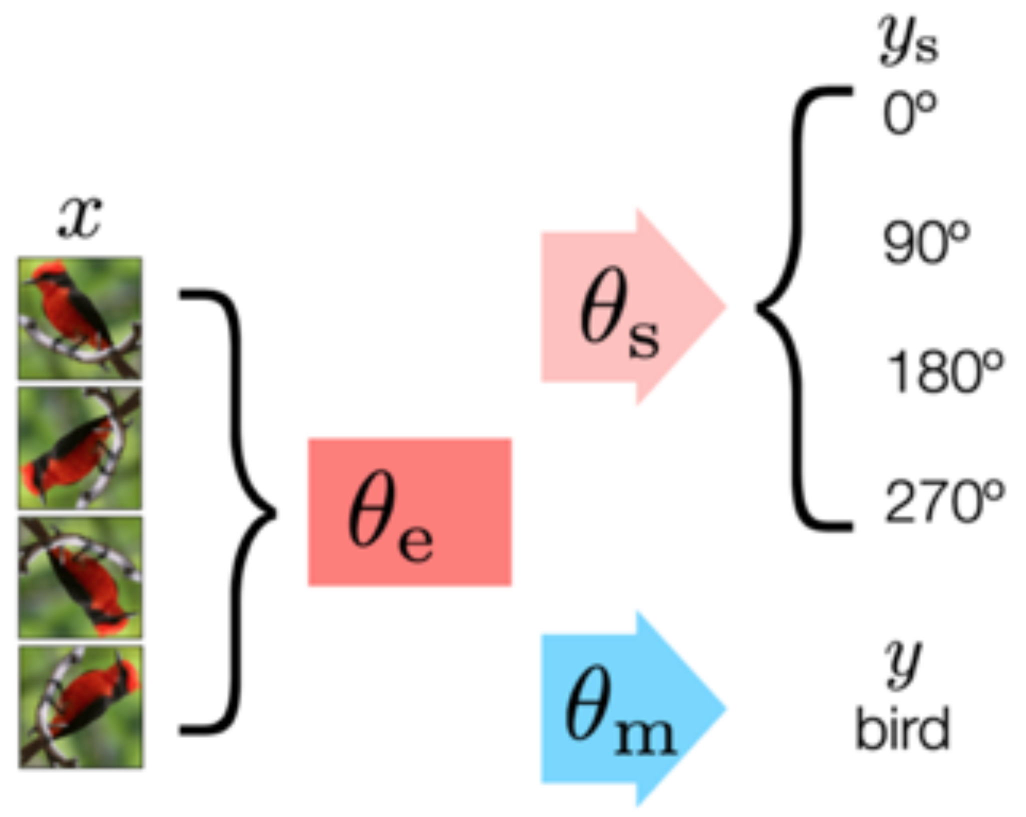
- Similar to subpopulation shift, **domain generalization** assumes several domains are provided at training time
- However, we typically do not assume that there is a domain imbalance issue that we must combat, e.g., via robustness
- Instead, we assume that we will be given new domains at test time, and our goal is to generalize to these new domains
- Sometimes, this problem setting is referred to as *zero-shot domain adaptation* or *multi-source domain adaptation*

Test time adaptation

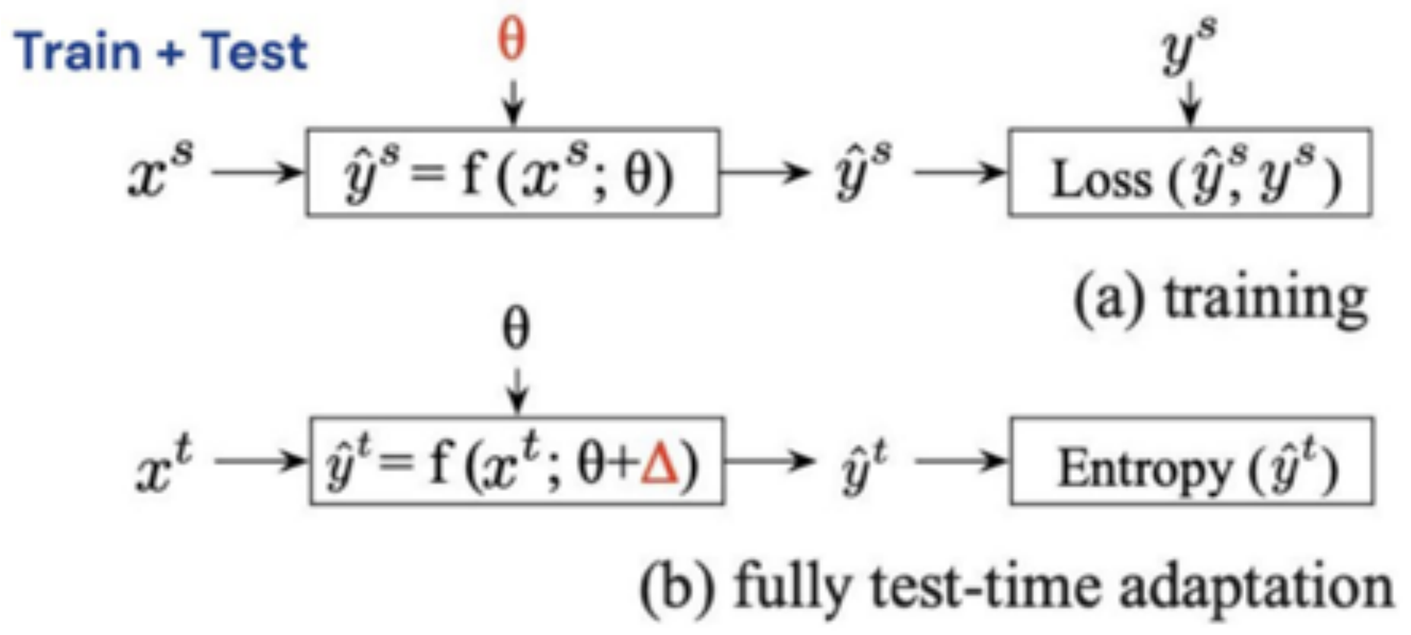
Self-supervised learning via:



BN adaptation (image from Nado et al, '20)



rotation prediction (Sun et al, ICML '20)

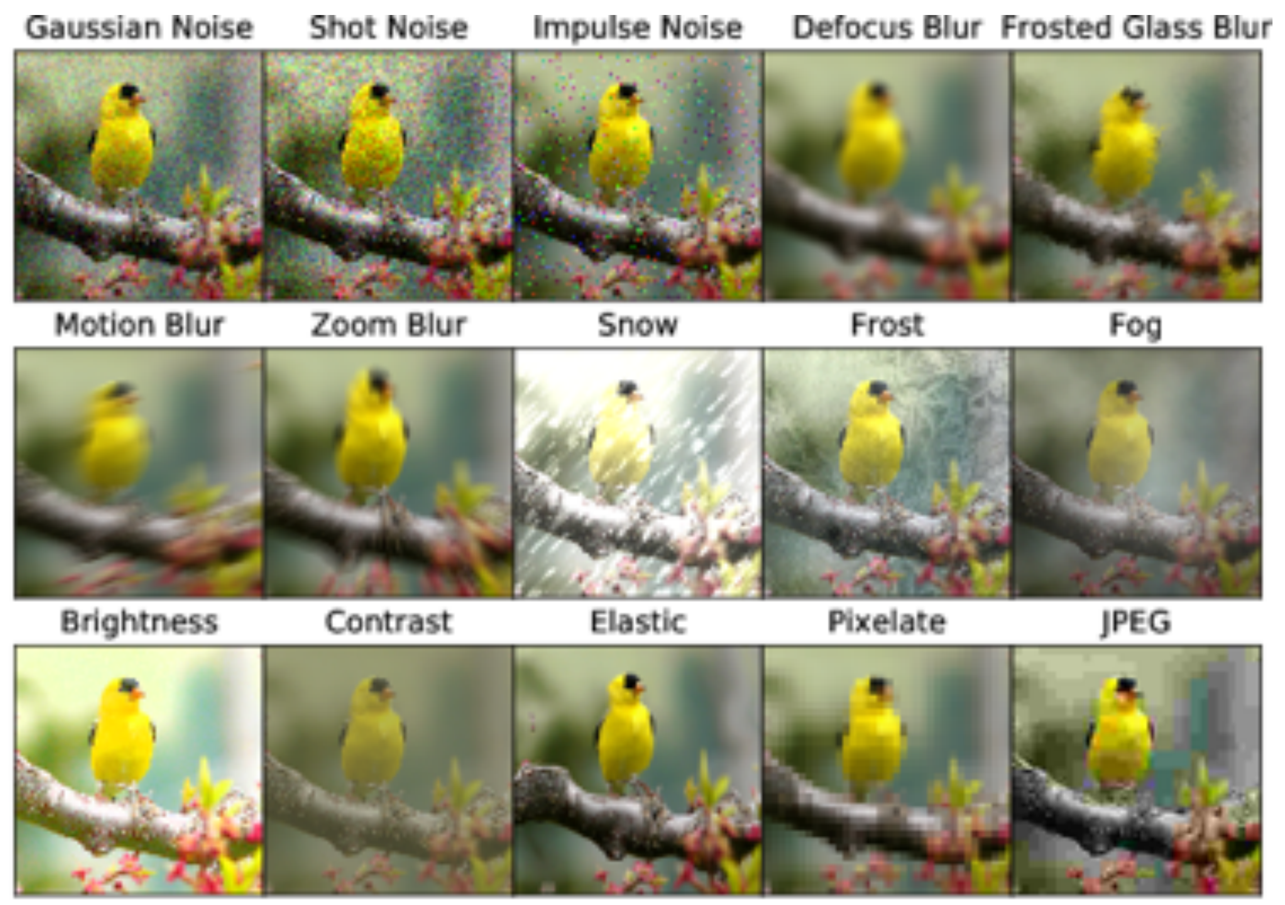


entropy minimization (Wang et al, ICLR '21)

"standard" model: $g : \mathscr{X} \to \mathscr{Y}$

adaptive model: $f : \mathscr{X} \times \mathscr{P}_{\mathbf{x}} \to \mathscr{Y}$

in practice, approximate $\mathscr{P}_{\mathbf{x}}$ with $(\mathbf{x}_1, \ldots, \mathbf{x}_K)$



Adversarial robustness

A threat model

- A simple threat model is to assume the adversary has an ℓ_p attack *distortion budget* ϵ, i.e., for some assumed p and ϵ, $\|\mathbf{x}_{\text{adv}} - \mathbf{x}\|_p \leq \epsilon$
- Not all distortions have a small ℓ_p norm, e.g., rotations — this simplistic threat model is common because it is a more tractable subproblem
- The adversary's goal is usually to find a distortion δ that maximizes the loss subject to its budget: $\mathbf{x}_{\text{adv}} = \mathbf{x} + \arg\max_{\delta:\|\delta\|_p \leq \epsilon} \ell(\theta; \mathbf{x} + \delta, y)$
- Review the details for both the FGSM and PGD attacks

Adversarial training (AT)

- The best way (we know of) to robustify models to ℓ_p attacks is **adversarial training (AT)**
- A common AT procedure is as follows:

  Sample minibatch $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(B)}, y^{(B)})$ from the training set

  Create $\mathbf{x}^{(i)}_{\text{adv}}$ (e.g., $\mathbf{x}^{(i)}_{\text{PGD}}$) from $\mathbf{x}^{(i)}$ for all i

  Optimize the average training loss on these adversarial training examples
- This does come with some downsides: currently, AT can reduce accuracy on non adversarial ("clean") examples by 10%+

Transferability of attacks

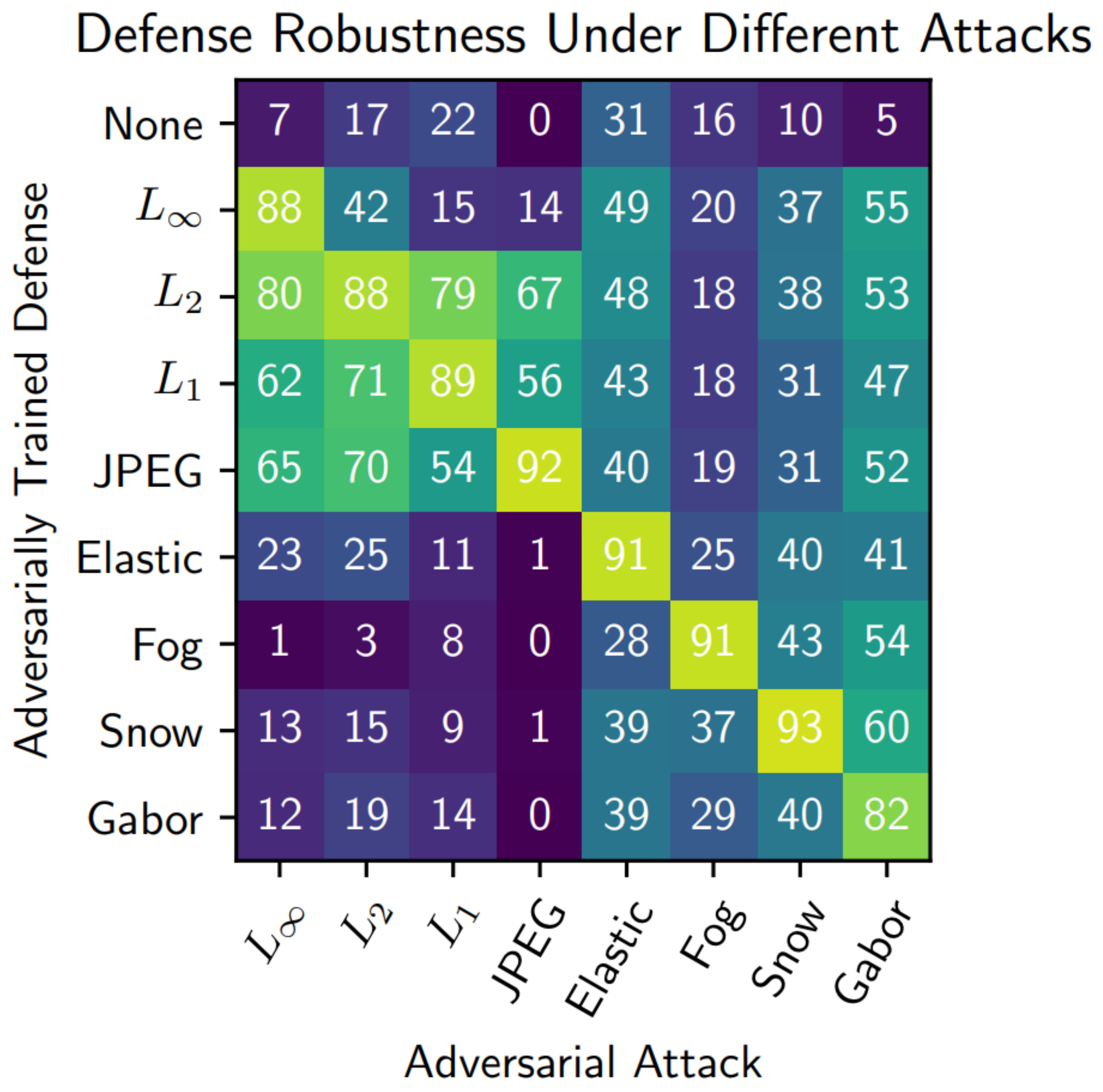
- An adversarial example crafted for one model can potentially be used to attack many different models
- Given neural network models M_1 and M_2, $\mathbf{x}_{\text{adv}}$ designed for M_1 sometimes also results in a high loss for $M_2(\mathbf{x}_{\text{adv}})$, even if M_2 is a different architecture
- Transfer rates can vary greatly, but even moderate amounts of transferability demonstrate that adversarial failure modes are somewhat shared across models
- Consequently, an attacker does not always need access to a model's parameters or architectural information in order to try and attack it

What improves adversarial robustness?

- Using larger and more diverse data
- Data augmentation, e.g., Cutout and CutMix
- Architectural choices such as using GELUs rather than ReLUs
- Be familiar with the details of all of the above

Unforeseen adversaries

- In practice, attackers could use unforeseen or novel attacks whose specifications are not known during training
- Models are far less robust to attacks they have not trained against, even if they have trained against other attacks
- To estimate robustness to unforeseen attacks, we should measure robustness to multiple attacks not encountered during training



	L_∞	L_2	L_1	JPEG	Elastic	Fog	Snow	Gabor
None	7	17	22	0	31	16	10	5
L_∞	88	42	15	14	49	20	37	55
L_2	80	88	79	67	48	18	38	53
L_1	62	71	89	56	43	18	31	47
JPEG	65	70	54	92	40	19	31	52
Elastic	23	25	11	1	91	25	40	41
Fog	1	3	8	0	28	91	43	54
Snow	13	15	9	1	39	37	93	60
Gabor	12	19	14	0	39	29	40	82

Deep Unsupervised Learning

Density/distribution modeling

Some potential motivations

- Why might we be interested in trying to model the data distribution?
- **Generation**: synthesize new data points that are useful/interesting/pretty
    - **Conditional generation**: synthesize specific data points of interest
- **Density modeling**: understand/analyze the likelihood of various data points
    - Doesn’t work that well OOD, at least not yet, but still could be useful…
- **Representation learning** (or *compression*, or *dimensionality reduction*, etc.): convert high dimensional data into lower dimensional representations

GANs summary

- GANs are the go-to model for image generation
    - Other types of models are catching up in terms of generated quality, however, they currently have other downsides such as being much slower to generate
- However, GANs are not density models, and there is not really a way to obtain probability estimates of data points from a GAN
    - Consequently, this makes GANs harder to evaluate as well — how can we say whether one GAN is better than another GAN (or some other model)?
    - Metrics for evaluating GANs usually rely on inputting generated images into a pretrained classifier (e.g., *Inception* and *FID* scores), but this is contentious

Autoregressive models summary

- Autoregressive models offer "best in class" modeling performance, oftentimes both qualitatively in terms of generation and quantitatively in terms of likelihood metrics
- This is true beyond images — there are very good autoregressive models for language (we already knew this) and audio (e.g., a somewhat dated example is WaveNet)
- Similar to GANs, we can modify autoregressive models to do conditional generation
- GANs are still superior when it comes to image generation
- Generating from autoregressive models is also very slow, comparatively speaking
- Lastly, autoregressive models do not naturally provide a notion of a latent space, so they are not used for representation learning

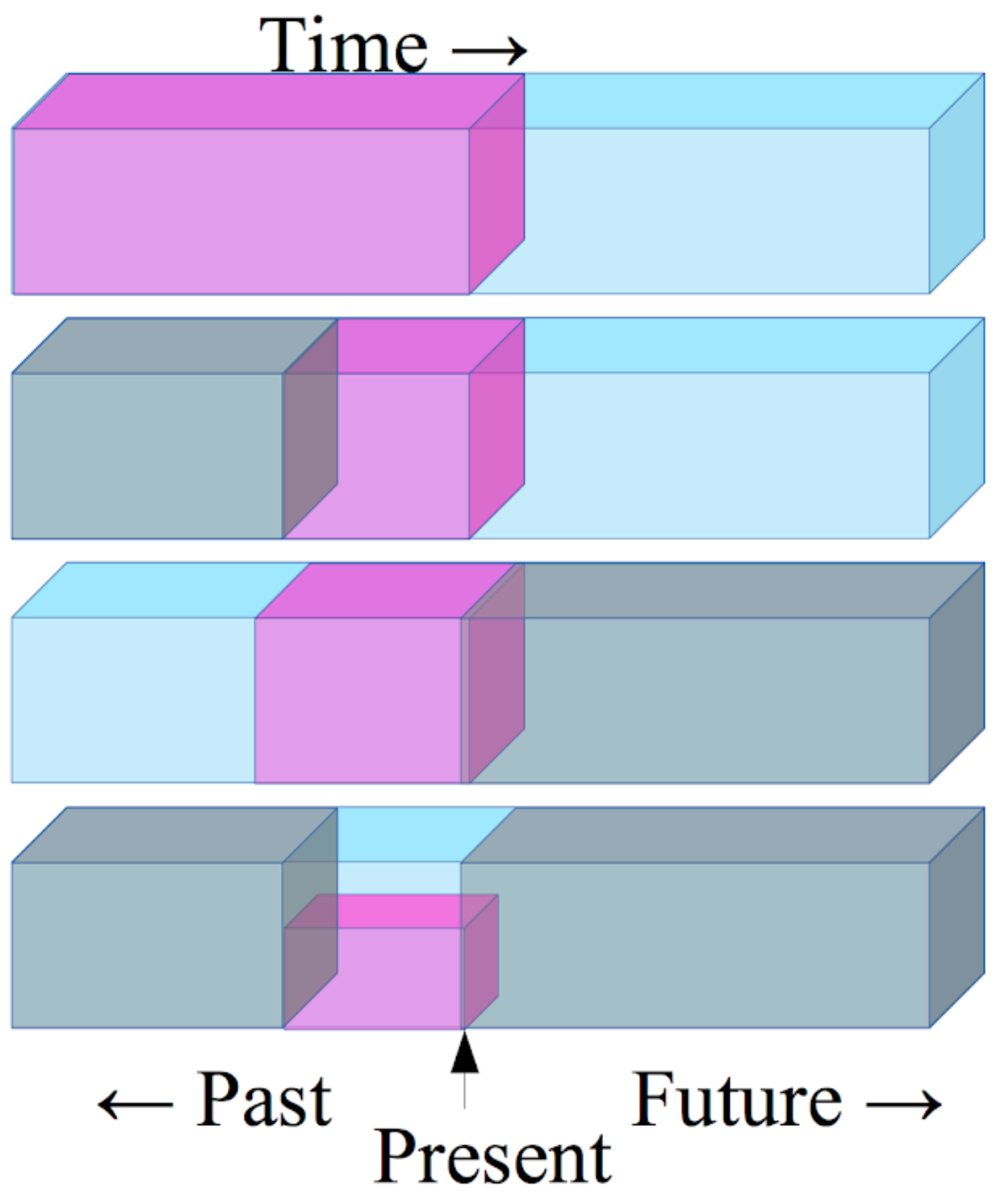
VAEs summary

- VAEs provide natural mechanisms for both representation learning and generation
    - Though estimating $p_\theta(\mathbf{x})$ is difficult, a lower bound can easily be obtained
- However, there is typically a tradeoff involved between representation learning and generation quality
- The VAEs which synthesize the best data points and result in the best (lower bounds of) likelihoods utilize complex priors and modeling choices, e.g., quantization and multiple levels of latent variables
    - This can make extracting useful representations more difficult

What is self-supervised learning?

According to Prof. Yann LeCun

- **Predict any part of the input from any other part.**
- **Predict the future from the past.**
- **Predict the future from the recent past.**
- **Predict the past from the present.**
- **Predict the top from the bottom.**
- **Predict the occluded from the visible**
- **Pretend there is a part of the input you don't know and predict that.**



Contrastive learning

- Several self-supervised learning methods are based on **contrastive learning**: learned representations should be close together for "similar" inputs and far apart for "dissimilar" inputs (we will define "similar" and "dissimilar" shortly)
- The most common contrastive learning loss is $-\log \frac{\exp\{\mathbf{z}^\top \mathbf{z}_+/\tau\}}{\sum_{i=1}^{K} \exp\{\mathbf{z}^\top \mathbf{z}_i/\tau\}}$
- Be familiar with CPC, MoCo, SimCLR, and CLIP at a high level

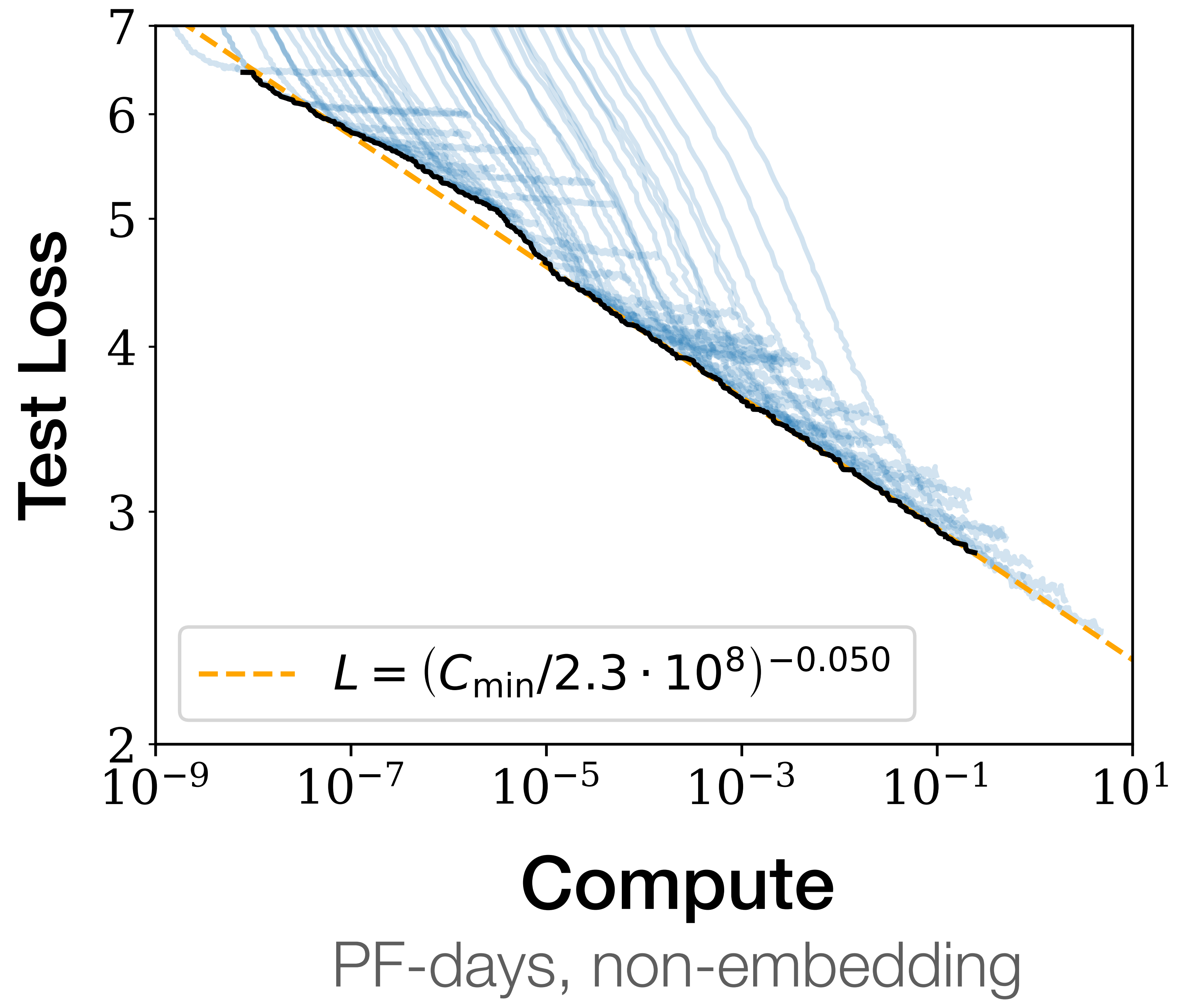
Masked autoencoding

- Randomly masking out parts of the input (e.g., 15% of tokens or 75% of image patches) and predicting these parts is a very effective self-supervised approach
- MAE vision transformers and BERT are both primarily transformer *encoders* that turn (masked) inputs into representations that are useful for downstream tasks
- During training, they are equipped with simple *decoders*, e.g., token classifiers for BERT and a small transformer (but not a transformer decoder) for MAE
    - These decoders attempt to recover the original input that was masked out, and the encoder is thus trained to produce useful contextual representations

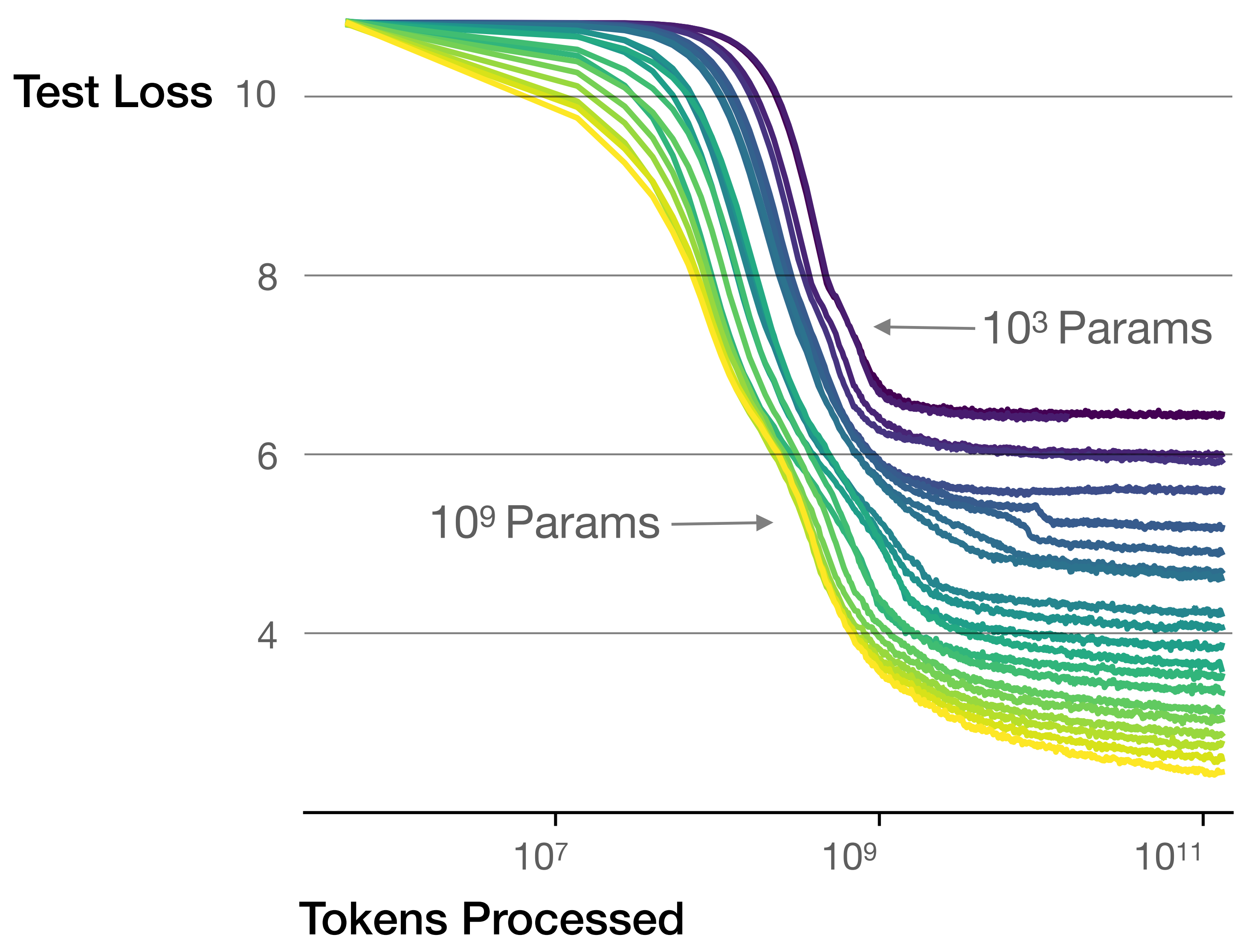
Massive models + the last guest lectures

Scaling laws for neural language models

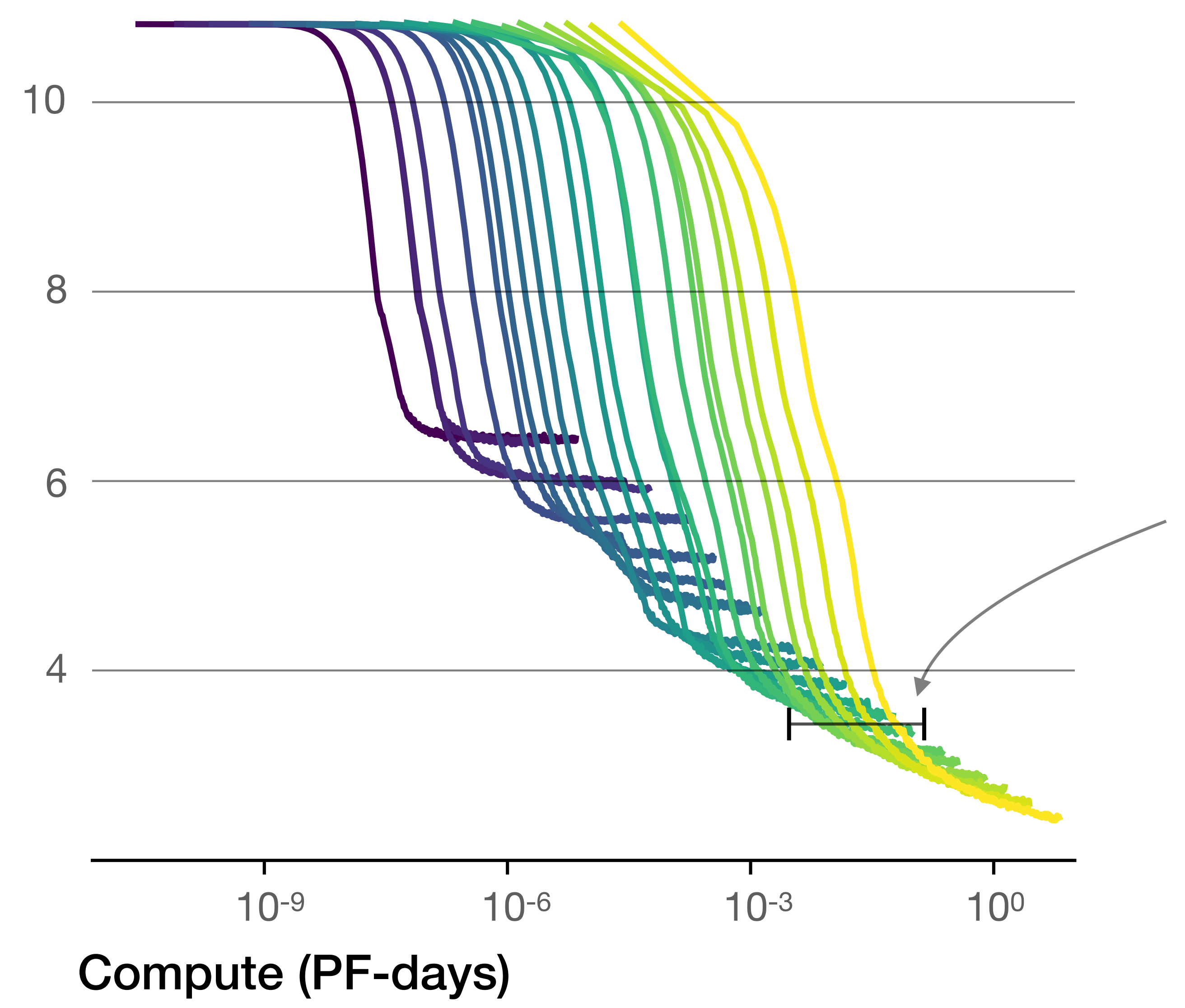
Kaplan et al, 2020



Larger models require **fewer samples** to reach the same performance



The optimal model size grows smoothly with the loss target and compute budget



Massive models

- Be familiar with GPT-3, Gopher, Chinchilla, Megatron-Turing NLG, and PaLM
- What did we learn from these different models? (maybe not much, from some)
- What is similar about these models? What is different?
- How do these models typically solve new tasks?
- Compare and contrast: few-shot prompting, chain-of-thought prompting, and “traditional” fine tuning

Potential harms and biases

- Papers about large models now typically come with a *model card* describing its details and intended uses, along with some analysis about potential harms
- For example, GPT-3 was analyzed for gender, race, and religion biases, and this shed light on its predispositions that (unfortunately) seem in line with its training
    - This analysis has also been carried out for the three other models mentioned
- Analysis on Gopher (and, to an extent, PaLM) demonstrates that large models, when given a *toxic* prompt, are more likely to generate toxic continuations
- These concerns, and more, have to be carefully studied and mitigated before deploying such models into sensitive applications

Meta-learning guest lecture

- Understand the motivations behind meta-learning and few-shot learning
- Understand the meta-learning problem setup and assumptions
- Understand the differences between black-box, optimization-based, and nonparametric meta-learning
- Read through the case study for student feedback generation and understand the high level details

Deep reinforcement learning guest lecture

- Understand the main challenges in applying the concepts we’ve learned to reinforcement learning (RL) and control
- Understand the terminology, e.g., what is the difference between observation and state?
- Understand the general approaches of behavioral cloning and DAgger
- Understand, at a high level, the basics of Q-learning and offline RL, as well as why offline RL is an interesting/important problem to study

Good luck!