Across the Great Divide: from ML Theory to Practice

Aditya Krishna Menon

Google NYC

Google Research

Introduction

Google Research

Research Scientist at Google NYC

Working on machine learning algorithm design and analysis

Past lives:

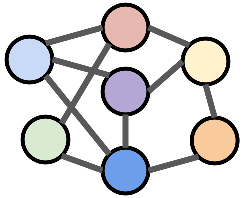
- USyd
- UCSD
- NICTA/CSIRO Data61/ANU

Supervised learning in theory

Google Research

Training data

Model training



Model predictions

0.2	0.1	0.4	0.2	0.1

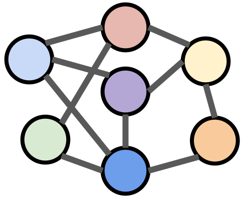
Supervised learning in theory

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

Supervised learning in practice

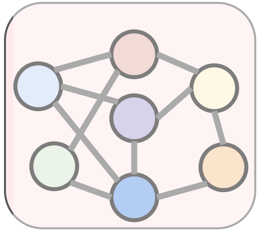
Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training

What if the model size is **too large**?



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

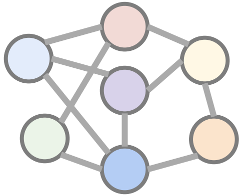
Supervised learning in practice

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

What if this loss is **expensive** to compute?

Model predictions

0.2	0.1	0.4	0.2	0.1

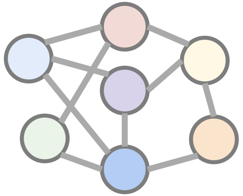
$$f(x^*)$$

Supervised learning in practice

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

What if this operation is **stochastic?**

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

Agenda

01 Background

02 Distillation

03 Extreme classification

04 Churn

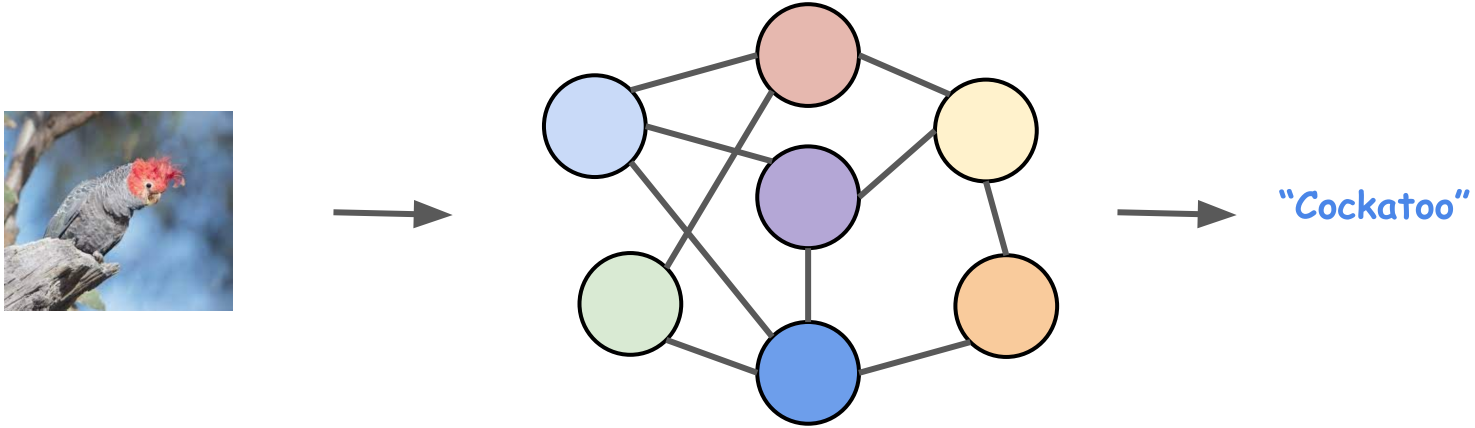
05 Summary

01

Background

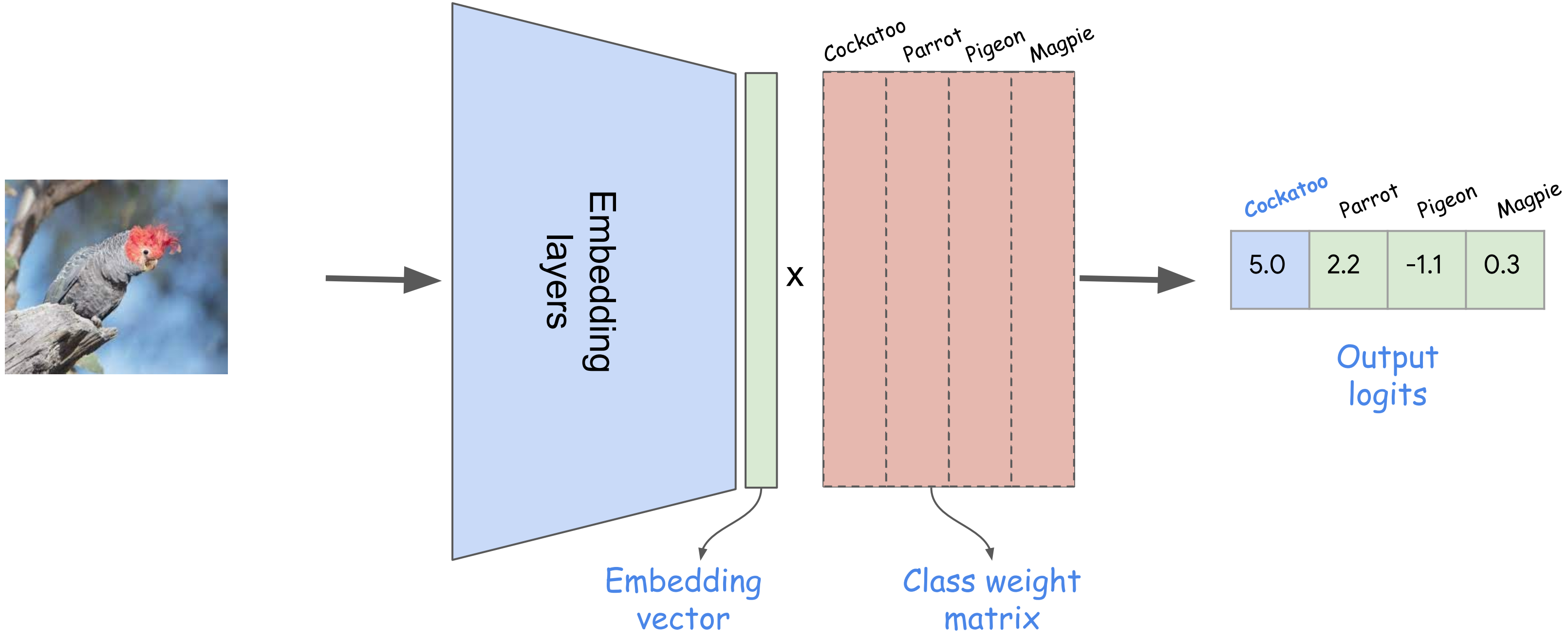
Neural networks for classification

Google Research



Neural networks for classification

Google Research



Neural networks for classification

Training objective: minimise **softmax cross-entropy**

Cockatoo	Parrot	Pigeon	Magpie
5.0	2.2	-1.1	0.3

$$\log \sum \exp \,[5.0,\ 2.2,\ -1.1,\ 0.3] \;-\; 5.0$$

This approximately minimises the (negative) **prediction margin**:

Parrot	Pigeon	Magpie
2.2	-1.1	0.3

$$\max \,[2.2,\ -1.1,\ 0.3] \;-\; 5.0$$

Highest score of "wrong" label

Neural networks for classification

Google Research

Training objective: minimise **softmax cross-entropy**

$\log \sum \exp$

Cockatoo	Parrot	Pigeon	Magpie
5.0	2.2	-1.1	0.3

\-

5.0

This equivalently minimises the **KL divergence:**

$$\mathrm{KL}(\mathbf{e}_y \,\|\, \mathbf{p}(x))$$

$$p_i(x) = \frac{\exp(s_i(x))}{\sum_{j \in [L]} \exp(s_j(x))}$$

"One-hot" label vector

1.0	0.0	0.0	...

"Softmax" probability vector

0.3	0.5	0.1	...

02

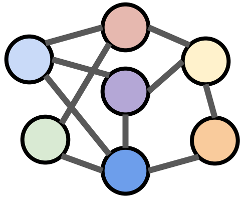
Distillation

Supervised learning in theory

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

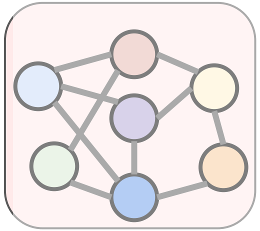
Supervised learning in practice

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training

What if the model size is **too large**?



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

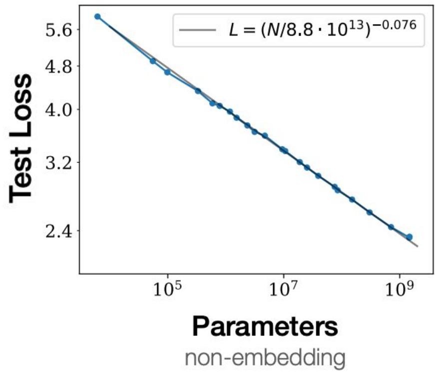
$$f(x^*)$$

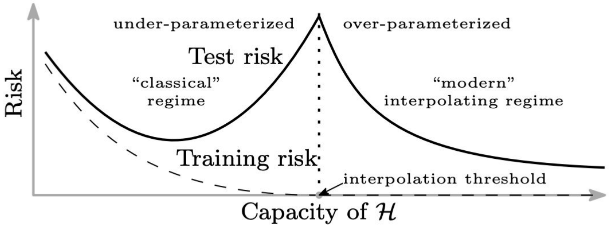
Why increase model size?

Google Research

😀 Can **work better**!

Particularly for complex tasks, e.g., language modelling





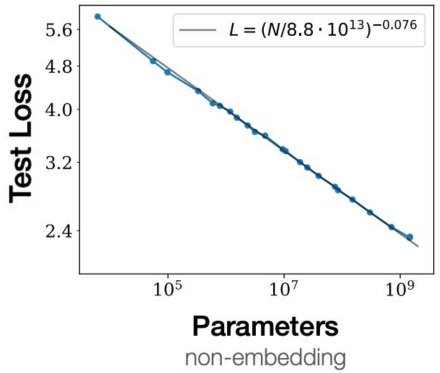
Belkin et al., '19. Reconciling modern machine learning practice and the bias-variance trade-off.
Kaplan et al. '20. Scaling Laws for Neural Language Models.

Why (not) increase model size?

Google Research

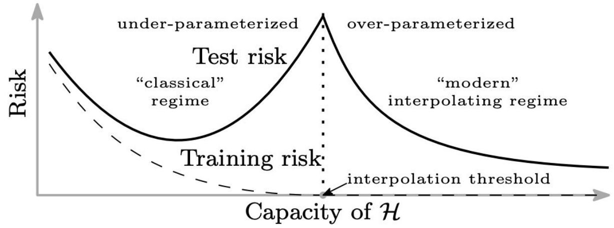
😀 Can **work better**!

Particularly for complex tasks, e.g., language modelling



😞 More expensive to **train**

😖 More expensive to **predict**

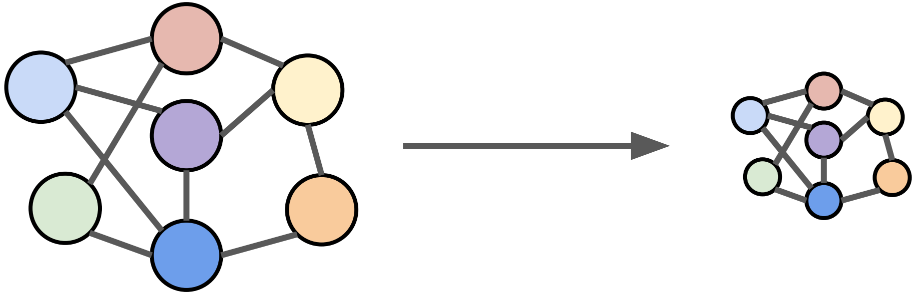


Belkin et al., '19. Reconciling modern machine learning practice and the bias-variance trade-off.
Kaplan et al. '20. Scaling Laws for Neural Language Models.

Idea: model compression

Google Research

Ideally, **compress** our model while **preserving** performance

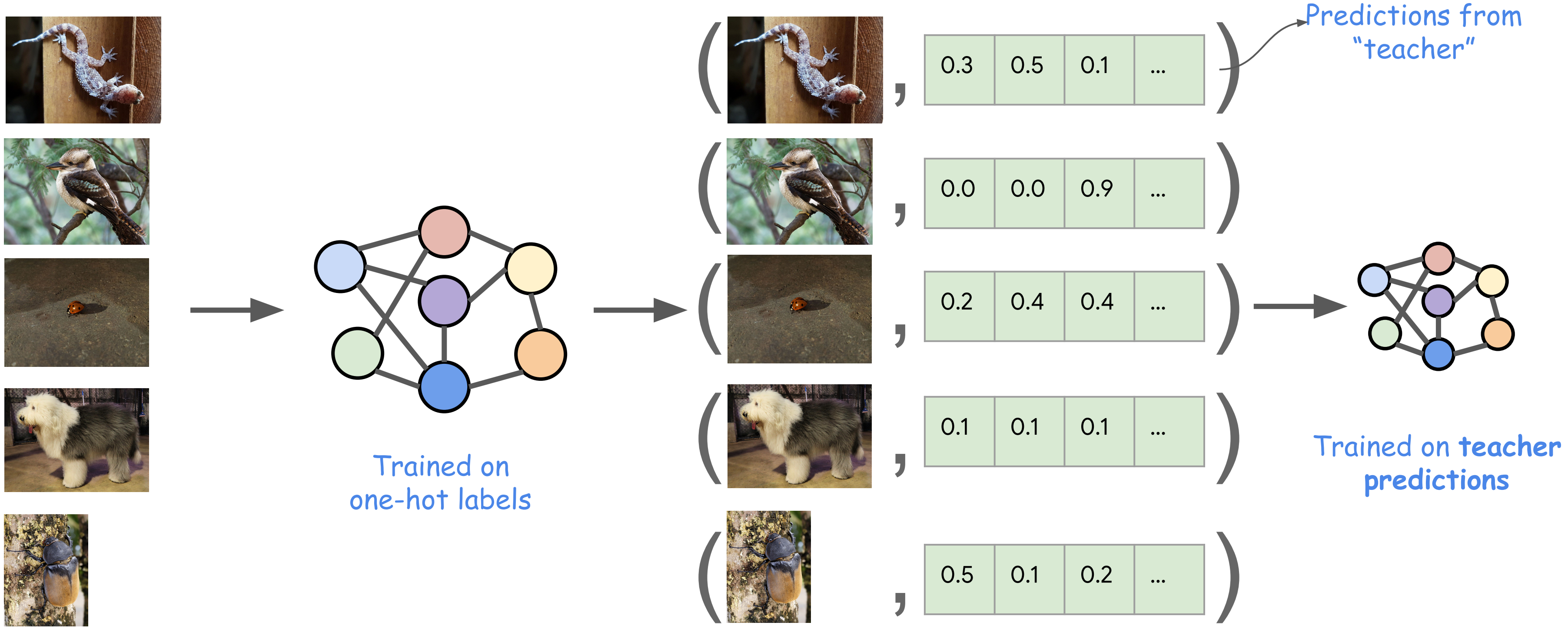


Many options: quantisation, architecture optimisation, **distillation**,.....

Distillation in a nutshell

Google Research

Train a “student” model using **soft predictions** from “teacher” model



Distillation loss function

Google Research

Minimise softmax cross-entropy

$$\log \sum \exp \begin{bmatrix} 5.0 & 2.2 & -1.1 & 0.3 \end{bmatrix} - \begin{bmatrix} 5.0 \end{bmatrix}$$

Cockatoo	Parrot	Pigeon	Magpie
5.0	2.2	-1.1	0.3

Distillation loss function

Google Research

Minimise **teacher-weighted** softmax cross-entropy

	Cockatoo	Parrot	Pigeon	Magpie			
$p^t($  $)$ x $\log \sum \exp$	5.0	2.2	-1.1	0.3	-	5.0	+
$p^t($  $)$ x $\log \sum \exp$	5.0	2.2	-1.1	0.3	-	2.2	+
$p^t($  $)$ x $\log \sum \exp$	5.0	2.2	-1.1	0.3	-	-1.1	+
$p^t($  $)$ x $\log \sum \exp$	5.0	2.2	-1.1	0.3	-	0.3	

Distillation loss function: formally

Google Research

Suppose the teacher's predictions are p^t

Then, we may minimise:

Input data

$$\frac{1}{N}\sum_{n=1}^{N}\left[(1-\alpha)\cdot \mathrm{KL}(\mathbf{e}_{y_n} \,\|\, p(x_n)) + \alpha \cdot \mathrm{KL}(p^{\mathrm{t}}(x_n) \,\|\, p(x_n))\right]$$

Mixing weight

1.0	0.0	0.0	...

"One-hot" label

0.3	0.5	0.1	...

"Soft" label

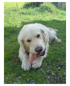
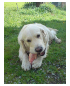
Why does distillation help?

Google Research

Transfers **class relationship** information

“Dark knowledge”

Learns which errors to penalise more

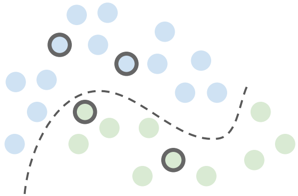
				
	0.8	0.2	0.0	0.0
	0.1	0.9	0.0	0.0
	0.0	0.1	0.8	0.1
	0.0	0.0	0.3	0.7

Per-sample **label smoothing**

Prevents over-confident predictions

Can be used on **unlabelled samples**

Form of semi-supervised learning!

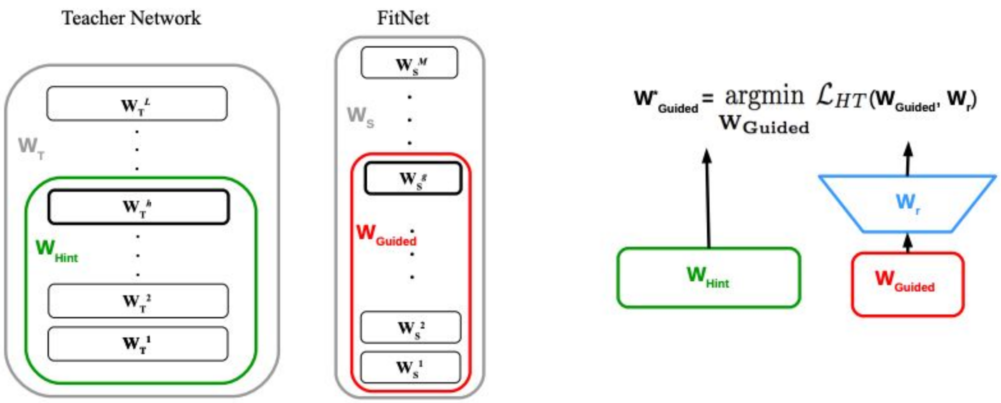


Beyond probability matching

Google Research

Can match **more structure** in teacher model

e.g., match embeddings, pairwise similarities, ...



(a) Teacher and Student Networks (b) Hints Training

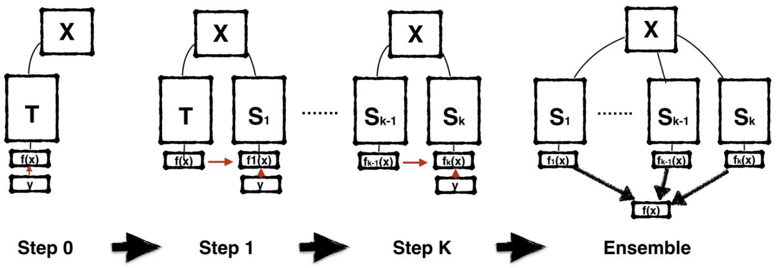
Romero et al., '15. FitNets: hints for thin deep nets.

Do we need complex teachers?

Google Research

No. You can "self-distill" **(!)**

Can give non-trivial gains



Why does this help?

Mostly an active area of research

One view: sample-dependent regularisation

Furlanello et al., '18. Born-again neural networks.

03

Extreme classification

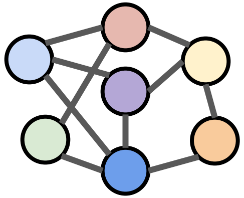
Supervised learning in theory

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

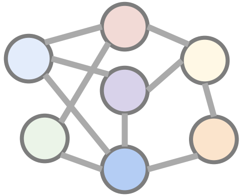
Supervised learning in practice

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

What if this loss is **expensive** to compute?

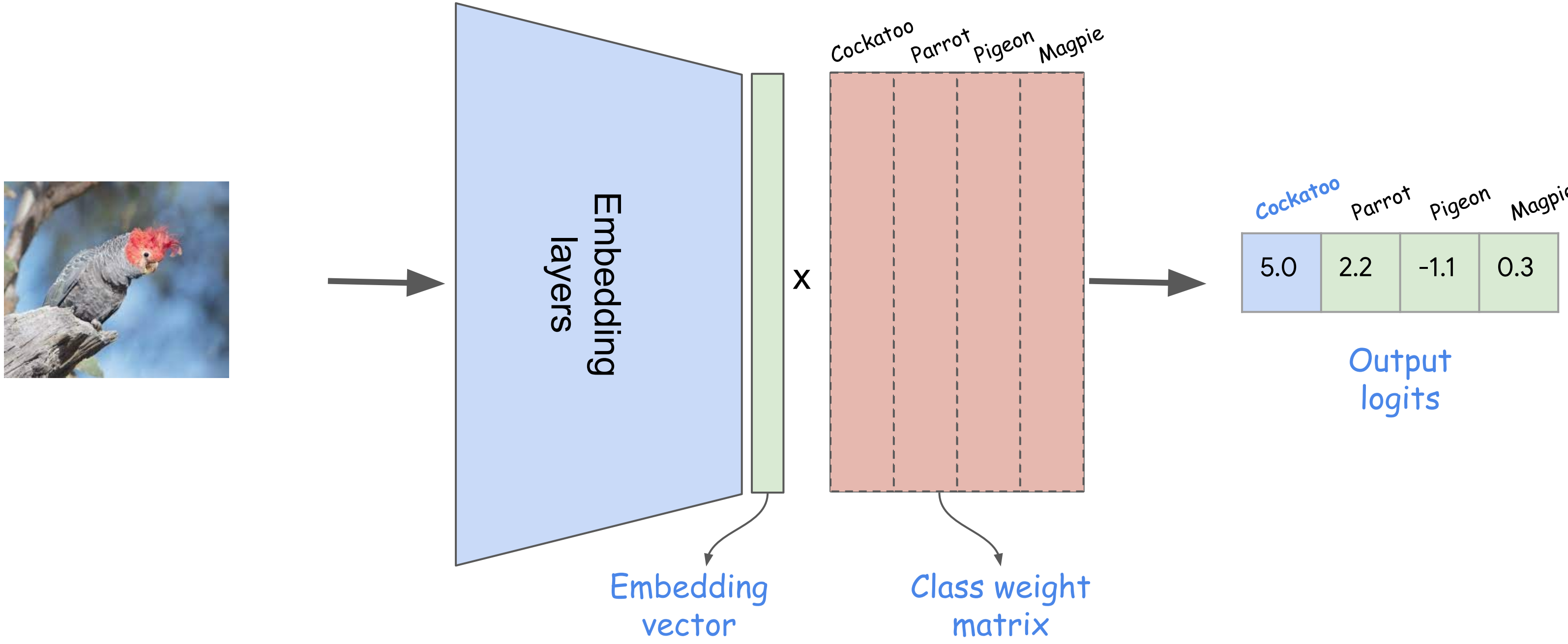
Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

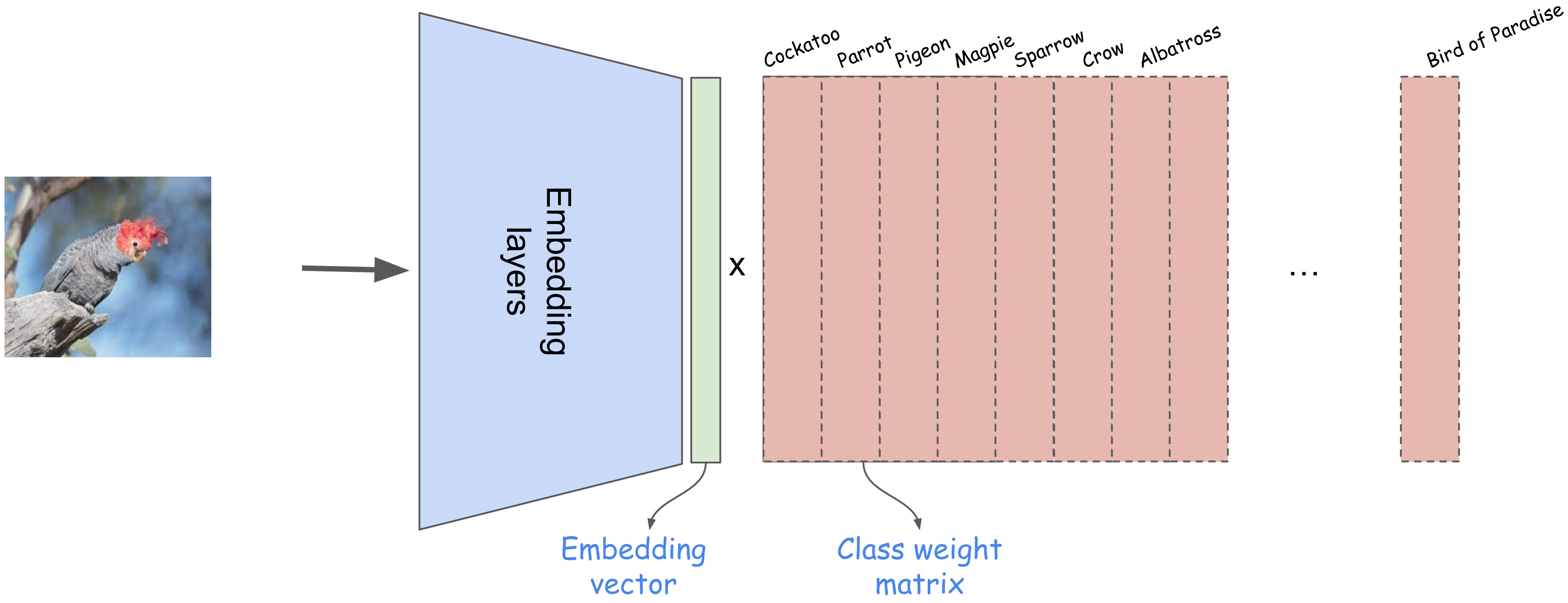
Neural networks for classification

Google Research



Neural networks for **extreme** classification

Google Research



Neural networks for extreme classification

Google Research

Training objective: minimise **softmax cross-entropy**

$$\log \sum \exp$$

Cockatoo	Parrot	Pigeon	Magpie	Sparrow	Crow	Albatross		Bird of Paradise
5.0	2.2	-1.1	0.3	0.1	-4.8	-1.9	...	0.5

\- 5.0

Hard to compute even for a single sample!

Negative sampling

Google Research

Select a subset of "**negative**" labels to contrast against "**positive**"

"Positive" label

"Negative" labels

Negative sampling

Google Research

Select a subset of “**negative**” labels to contrast against “**positive**”

“Positive” label

“Negative” labels

Ideally, we would like the sampling to:

- Be **easy** to compute
- Result in **informative** negatives

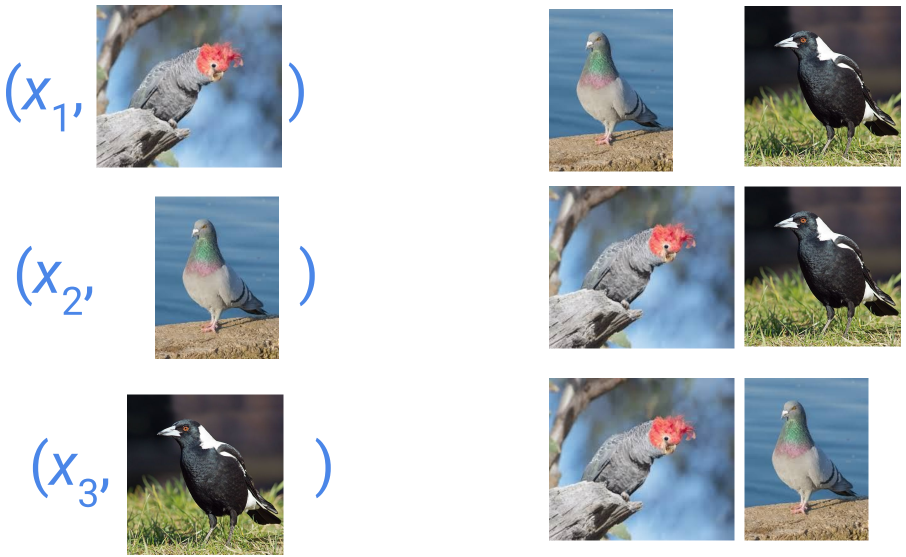
Choosing the sampling distribution

Google Research

Solution #1: within-batch negatives

"Positive" label

"Negative" labels



- Easy to compute
- Biased towards frequent labels

Choosing the sampling distribution

Google Research

Solution #2: uniform random negatives

"Positive" label

$(x_1,$  $)$

$(x_2,$  $)$

$(x_3,$  $)$

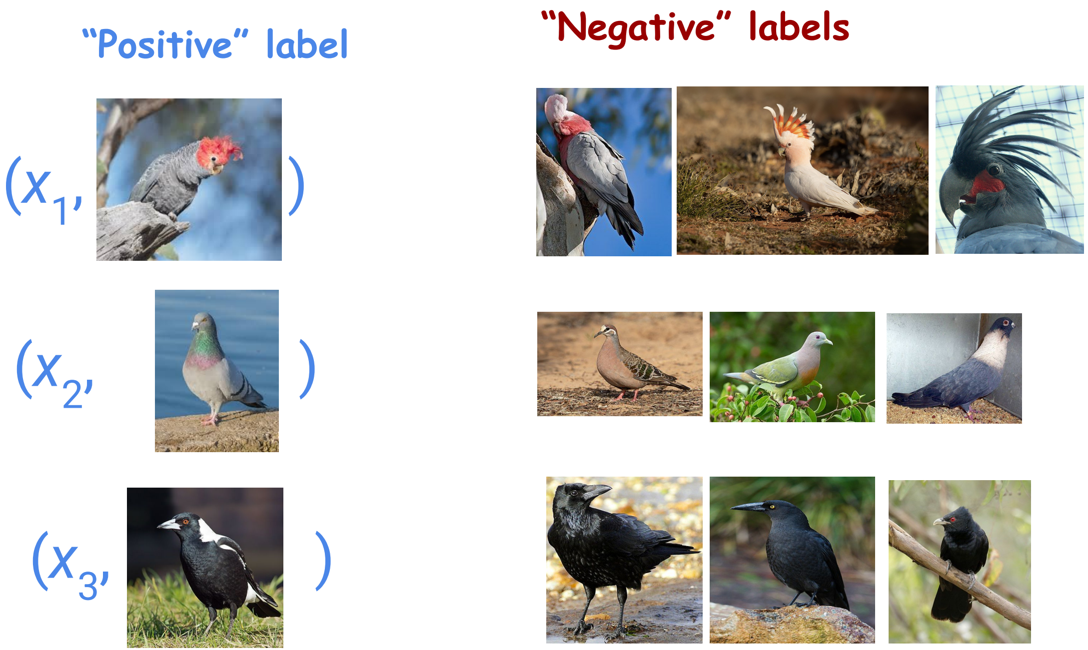
"Negative" labels

- 😀 Easy to compute
- 😀 Not biased towards any label
- 😞 May not be informative

Choosing the sampling distribution

Google Research

Solution #3: hard negative mining



- 😀 Maximally informative
- 😞 Hard to compute

Finding hard-negatives

Ideally, find labels that are **maximally confusing** for model

- 😞 this set changes as training progresses
- 😖 finding these exactly still requires sweeping over all labels!
- 😀 can **approximate**: find hardest labels **within a large batch** of uniformly sampled labels

$(x_1,$  $)$

Reddi et al., '18. Stochastic-negative mining for learning in large output spaces.

04

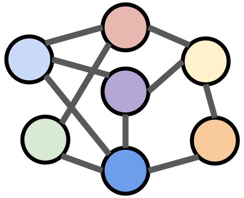
Model churn

Supervised learning in theory

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

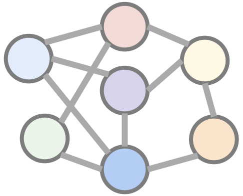
Supervised learning in practice

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

What if this operation is **stochastic?**

Model predictions

0.2	0.1	0.4	0.2	0.1

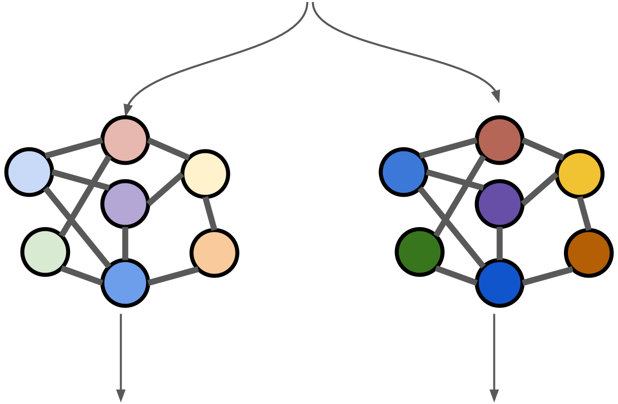
$$f(x^*)$$

Churn in a nutshell

Google Research

Model prediction disagreement under different training and/or inference conditions

Test example



Base model,
e.g., ResNet

Same model,
re-trained

Power drill
(probability 0.48)

Sewing machine
(probability 0.68)

Churn for classification

Google Research

Suppose we have two classification models, M_1 and M_2

e.g., two independently trained models on the same data

The corresponding churn is the probability of disagreement:

$$\text{Churn}(M_1, M_2) = \mathbf{Pr}(M_1(x) \neq M_2(x))$$

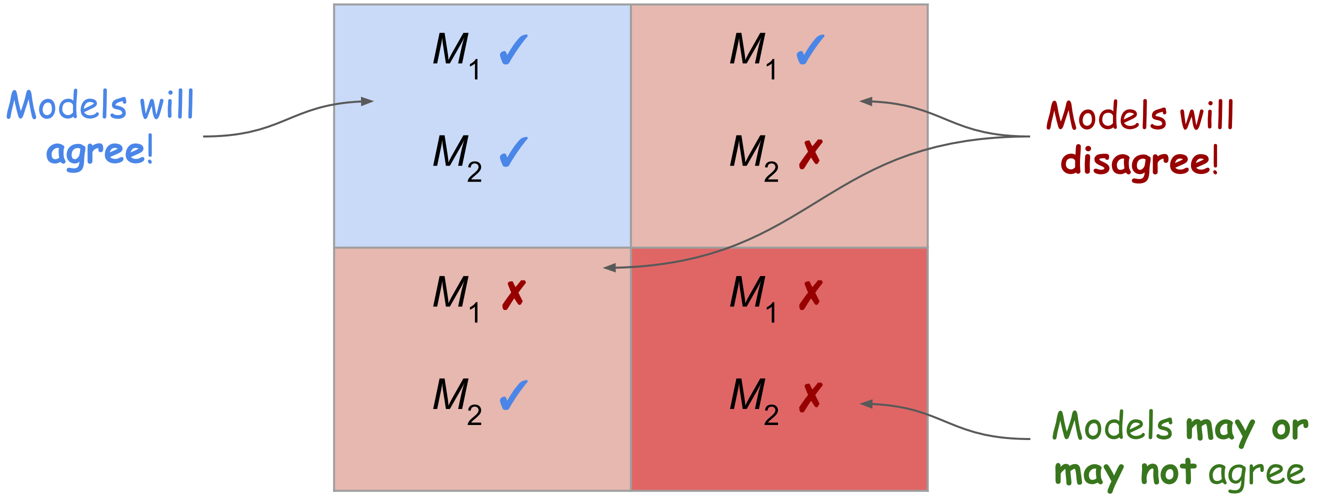
Fraction of times they predict a different label

Churn versus accuracy

Google Research

Churn can only occur when one or both models is wrong

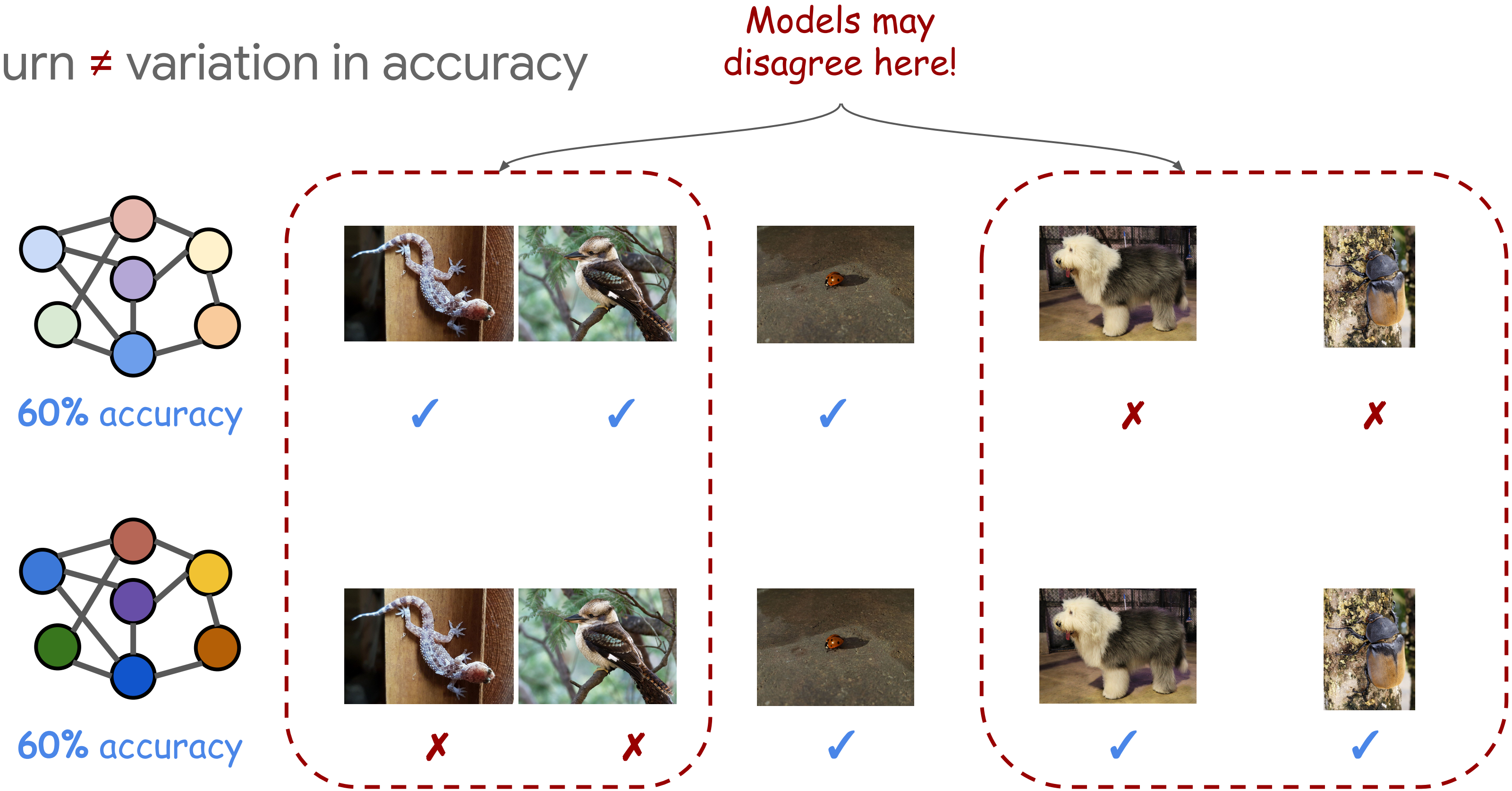
The better the individual models, the lower the churn



Churn versus accuracy variation

Google Research

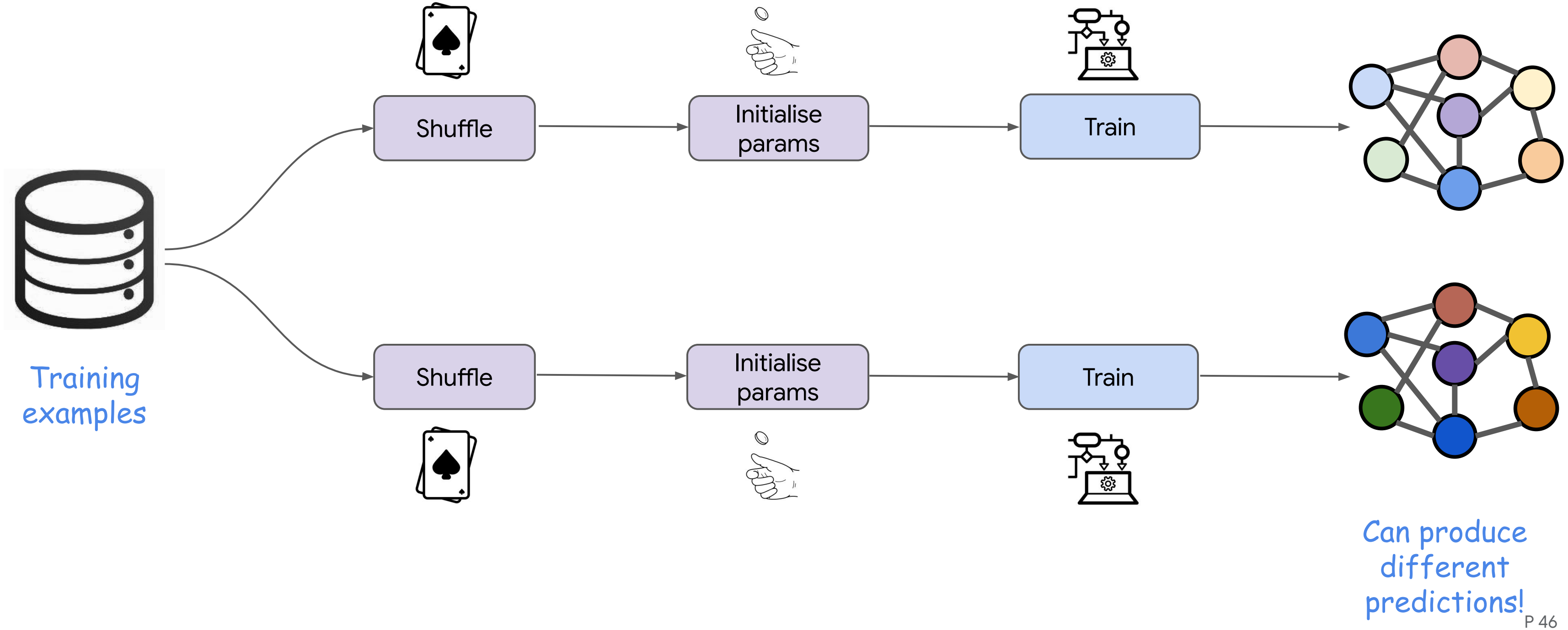
Churn ≠ variation in accuracy



Churn from model training

Google Research

Churn exists even when training on the **same** data, due to several sources of randomness:

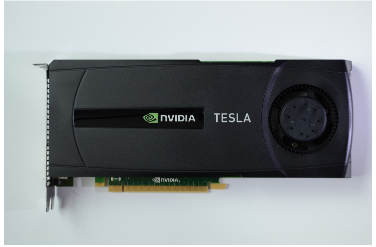


Churn from computing platform

Google Research

Inherent non-determinism in GPU and TPU

Floating-point addition is not associative!




```
[1] (0.1+0.2)+0.3

    0.6000000000000001

[2] 0.1+(0.2+0.3)

    0.6
```

Do neural models exhibit churn?

Google Research

Unfortunately, **yes**

Predictions from 5x independently trained ResNet models on ImageNet

76.0% accuracy with 0.1% standard deviation

Disagreement on **15%** of examples!

power drill - 0.48
sewing machine - 0.68
sewing machine - 0.28
sewing machine - 0.53
power drill - 0.87

wooden spoon - 0.24
spaghetti squash - 0.71
French loaf - 0.67
French loaf - 0.57
French loaf - 0.63

swing - 0.82
lawn mower - 0.56
tricycle - 0.49
balance beam - 0.75
lawn mower - 0.45

fountain pen - 0.46
can opener - 0.28
crossword - 0.62
hammer - 0.22
crossword - 0.5

How do we mitigate such prediction differences?

Bhojanapalli et al., ‘20. On the reproducibility of neural network predictions

Co-distillation

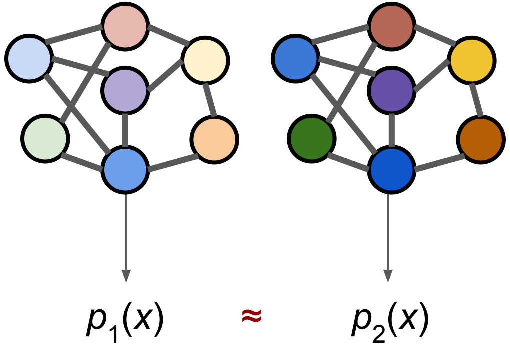
Google Research

Motivation: churn is partly a result of randomness in training

Idea: explicitly try to smooth out this randomness!

Approach: train two independent models, and encourage their predictions to be similar to each other

- Can be seen as “co-distillation”
- **Bonus:** also improves performance!



Anil et al., ‘18. Large scale distributed neural network training through online distillation

Distillation for churn

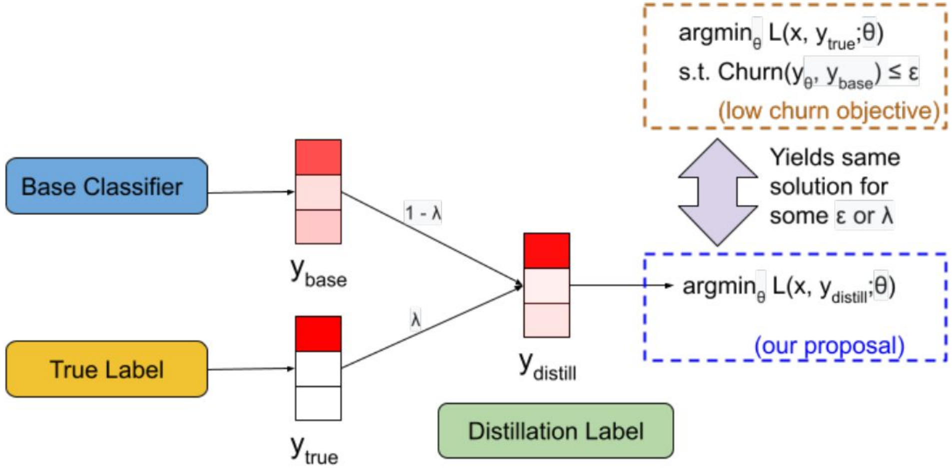
Google Research

Churn can also occur more generally between model versions

e.g., models trained on different weeks, with different architectures, ...

Idea: constrain predictions to be similar to original model

Implementation: distillation!



Jiang et al., '20. Churn Reduction via Distillation

05

Summary

Google Research

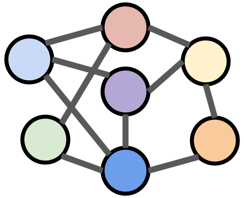
Supervised learning in practice!

Google Research

Training data

$$\{(x_n, y_n)\}_{n=1}^{N}$$

Model training



$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{n \in N} \ell(y_n, f(x_n))$$

Churn reduction

Negative mining

Distilled label

Model predictions

0.2	0.1	0.4	0.2	0.1

$$f(x^*)$$

Thank you!

Aditya Krishna Menon
Research Scientist
Google NYC

Google Research

Acknowledgements

Google Research

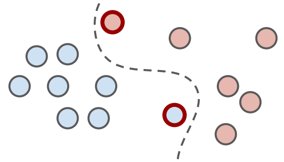
Images from Wikimedia Commons:

- Mediterranean house gecko, CC-BY-SA4.0
- Old English Sheepdog, CC-BY-SA4.0
- Dynastes Hercules, CC-BY-SA4.0
- Coccinellidae, CC-BY-SA4.0
- Major Mitchell's cockatoo, CC-BY-SA4.0
- Pink-necked green pigeon, CC-BY-SA4.0
- Common bronzewing pigeon, CC-BY-SA4.0
- Boeing 777-300ER, CC-BY-SA4.0
- Toyota Corolla, CC-BY-SA4.0
- Gang-gang cockatoo, CC-BY-SA3.0
- Laughing Kookaburra, CC-BY-SA3.0
- Australian magpie, CC-BY-SA3.0
- Paloma bravia, CC-BY-SA3.0
- Galah, CC-BY-SA3.0
- Egyptian swift pigeon, CC-BY-SA3.0
- Currawong, CC-BY-SA3.0
- Eastern koel, CC-BY-SA3.0
- Golden retriever, CC-BY-SA3.0
- Albatross, CC-BY-2.5
- Cat, CC-BY-2.5
- Rose-ringed parakeet, CC-BY-2.0
- Rock sparrow, CC-BY-2.0
- Carrion crow, CC-BY-2.0
- Greater bird of paradise, CC-BY-2.0
- Palm cockatoo, CC-BY-2.0
- nVidia Tesla, CC-0

Entropy regularisers

Google Research

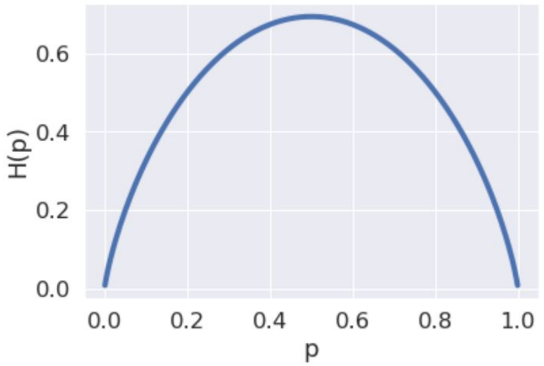
Motivation: churn occurs when samples' labels flip



Idea: move examples away from the classifier boundary!

Approach: reduce prediction entropy: for logits p, penalise

$$H(p) = -\Sigma_i \, p_i \log p_i$$



discourage highly uncertain predictions

Churn from data changes

Google Research

Refreshes of the data can change the learned model