Mastering Model Building

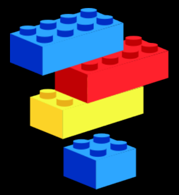
How to design (and debug) your ML model

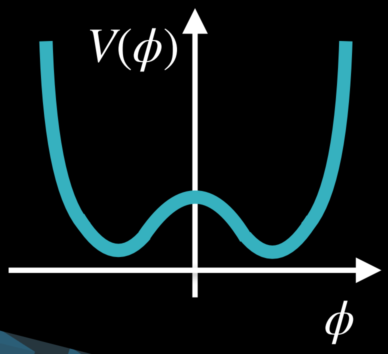
Nicole Hartman
nicole.hartman@tum.de

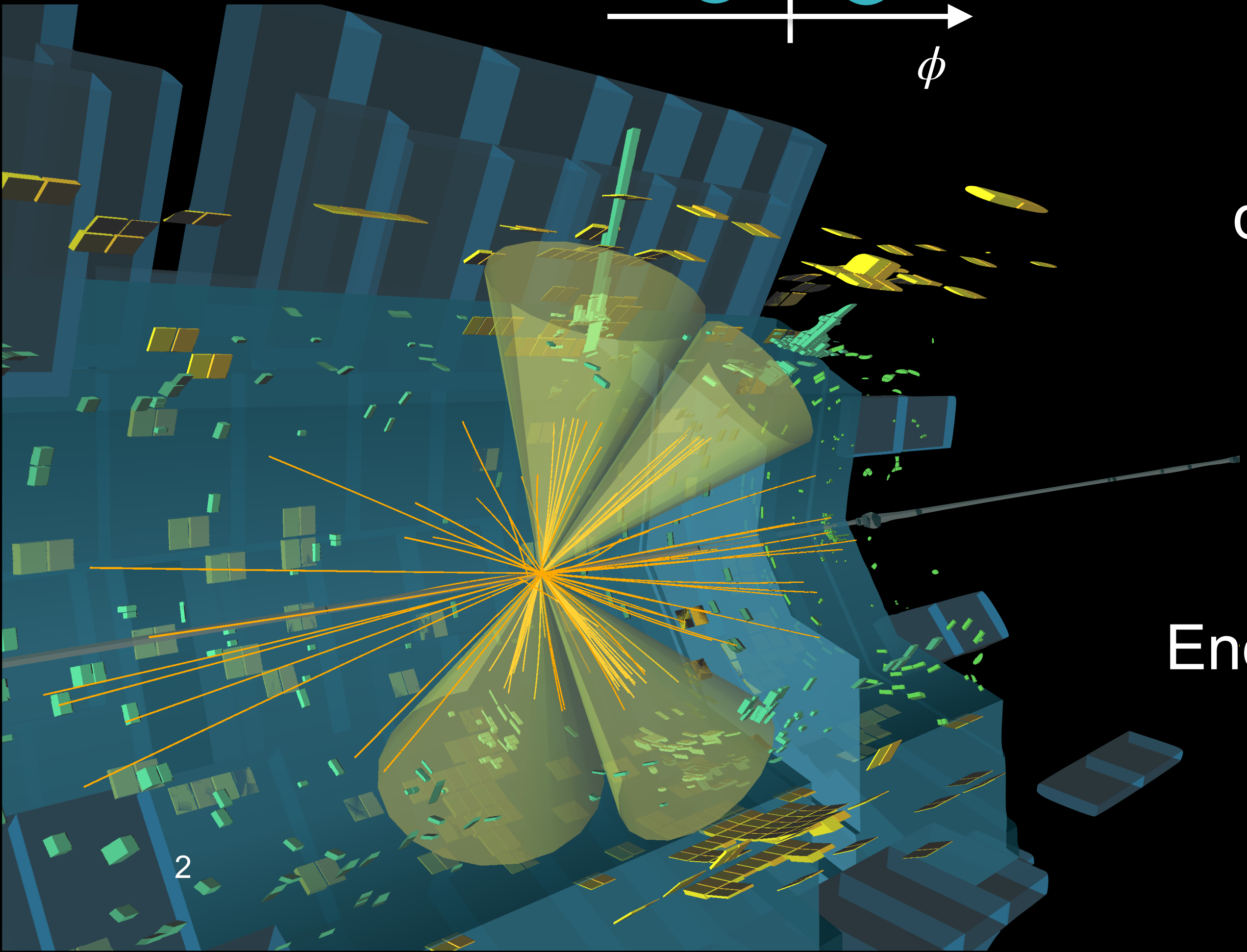
Train the Trainer workshop
17th Sept 2025

ODSL
ORIGINS Data Science Lab

Who am I? … and what got me into ML?







Sets and transformers for particle identification

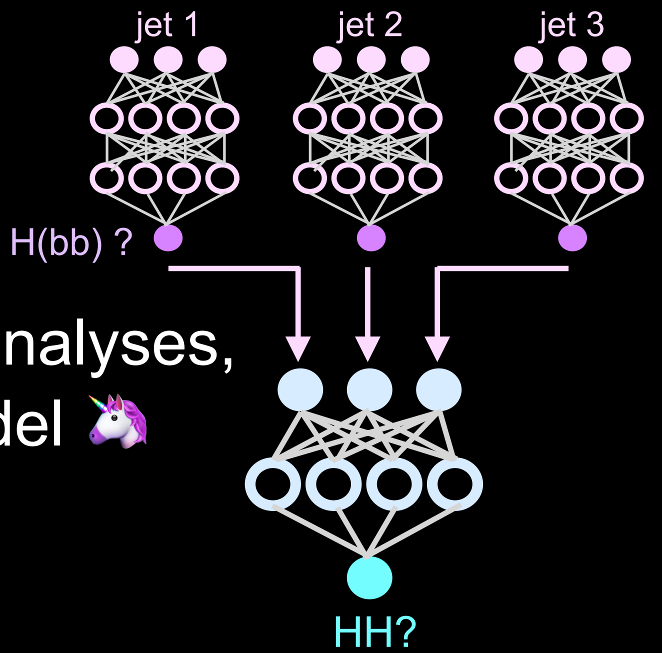
2505.19689, accepted to Nature Comm
FTAG-2025-01, latest results

Generative models / density ratio estimation

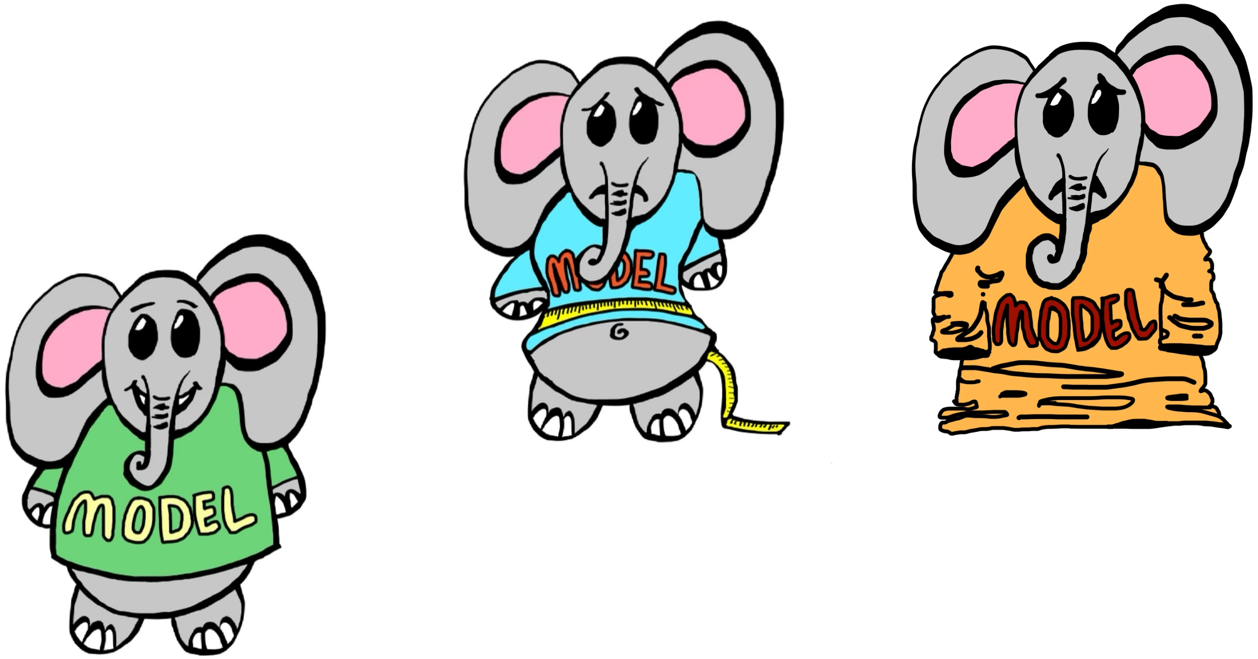
Ch 13 PhD thesis
Phys. Rev. D 108 (2023) 052003

End-to-end optimizable analyses, HEP's foundation model 🦄

MLST 5 025075 (2024)



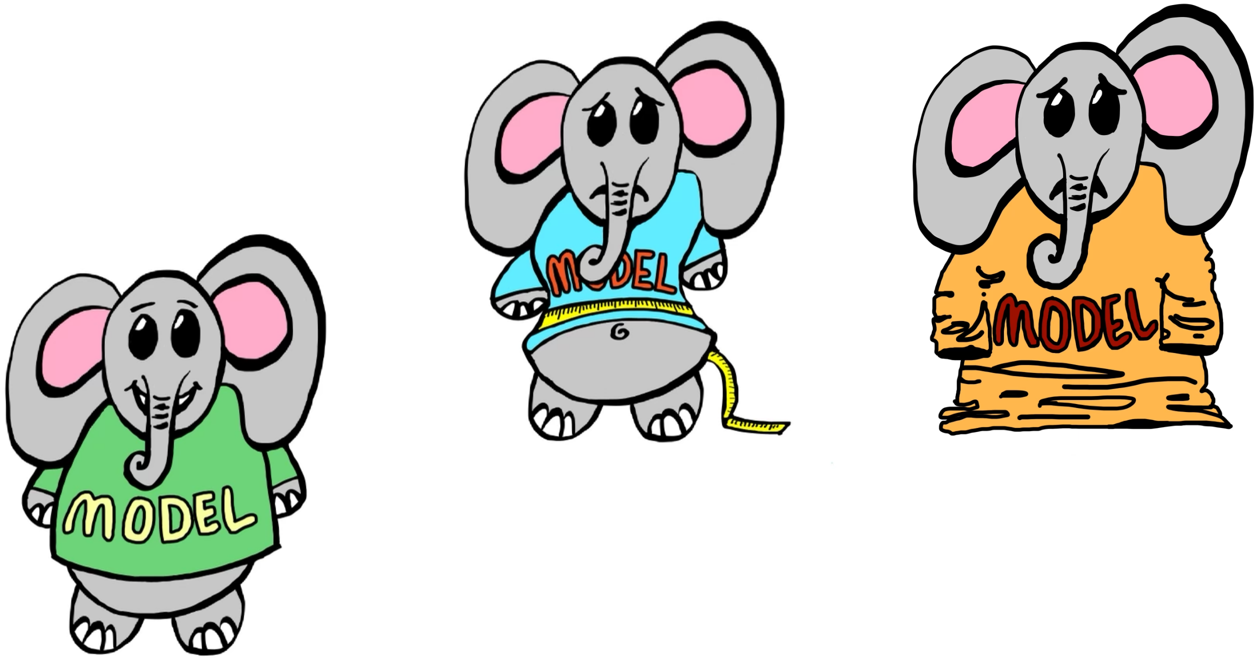
In the context of science, the well-known adage “a picture is worth a thousand words” might well be “a model is worth a thousand datasets”



Elephant artwork curtesy of B. Hartman

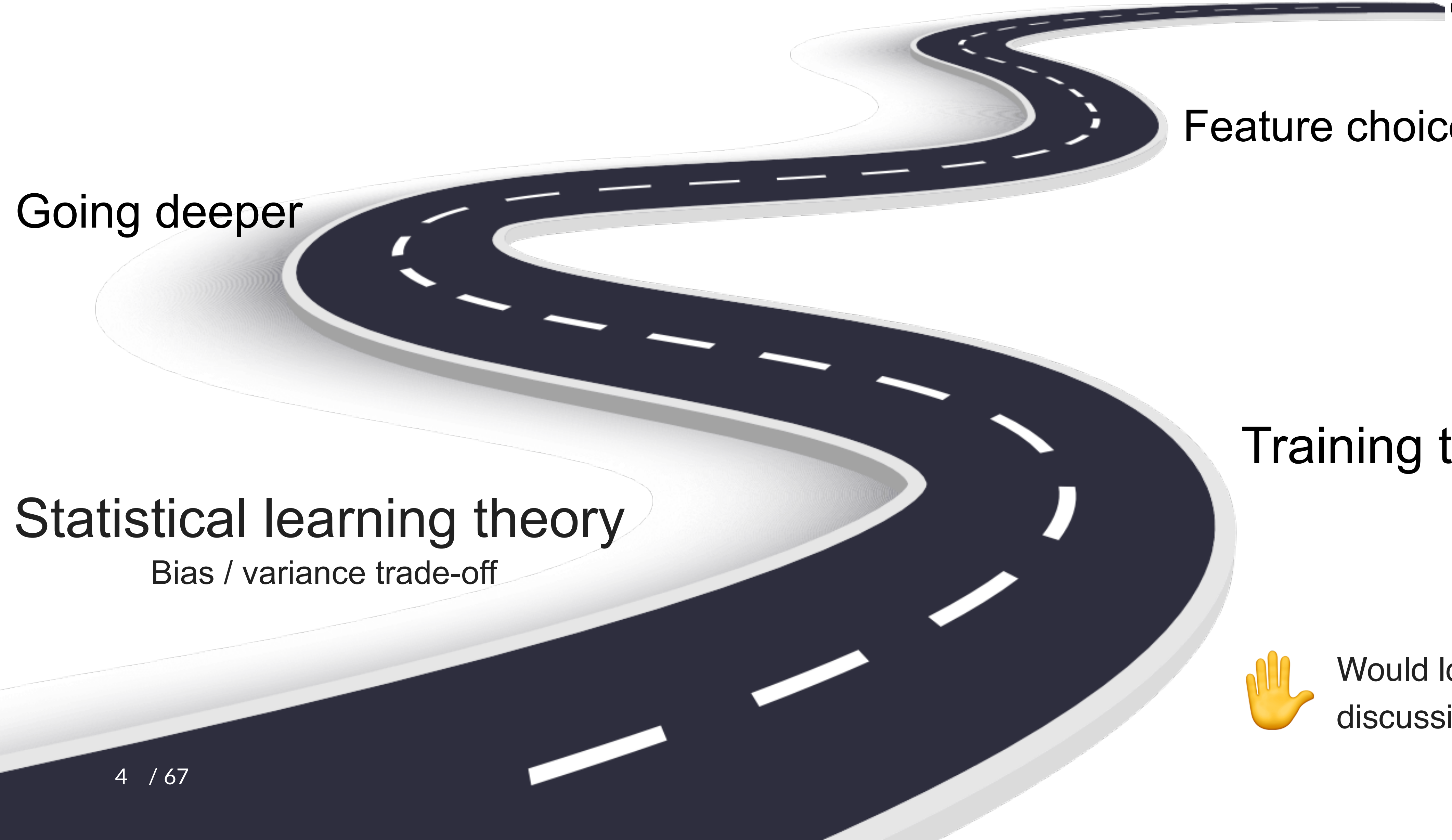
In the context of science, the well-known adage "a picture is worth a thousand words" might well be "a model is worth a thousand datasets"

This talk: How to choose this model



Elephant artwork curtesy of B. Hartman

What we'll cover today



Open question

Feature choices

Going deeper

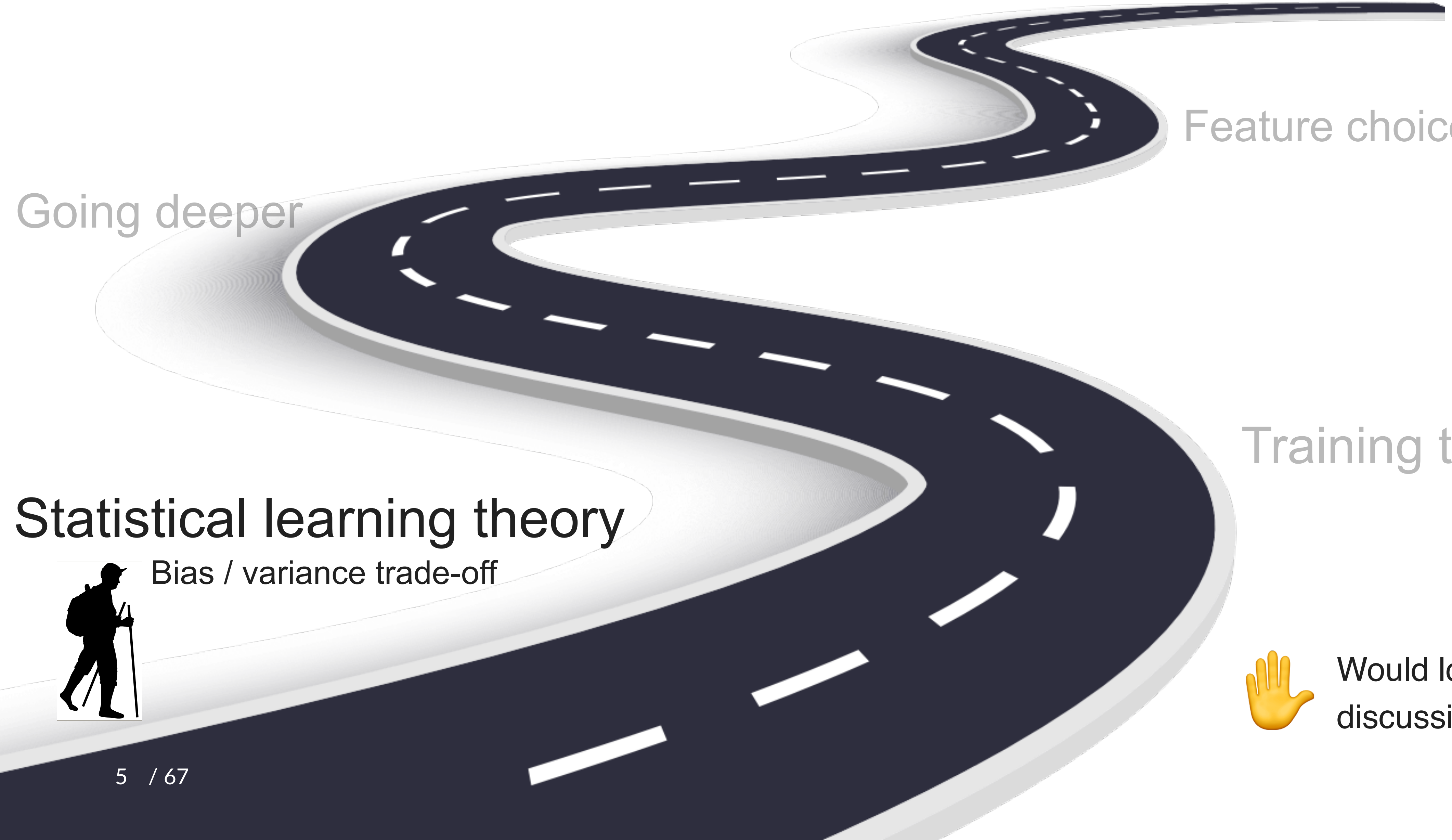
Training techniques

Statistical learning theory

Bias / variance trade-off

✋ Would love feedback + discussions!

Starting off...



Open question

Feature choices

Going deeper

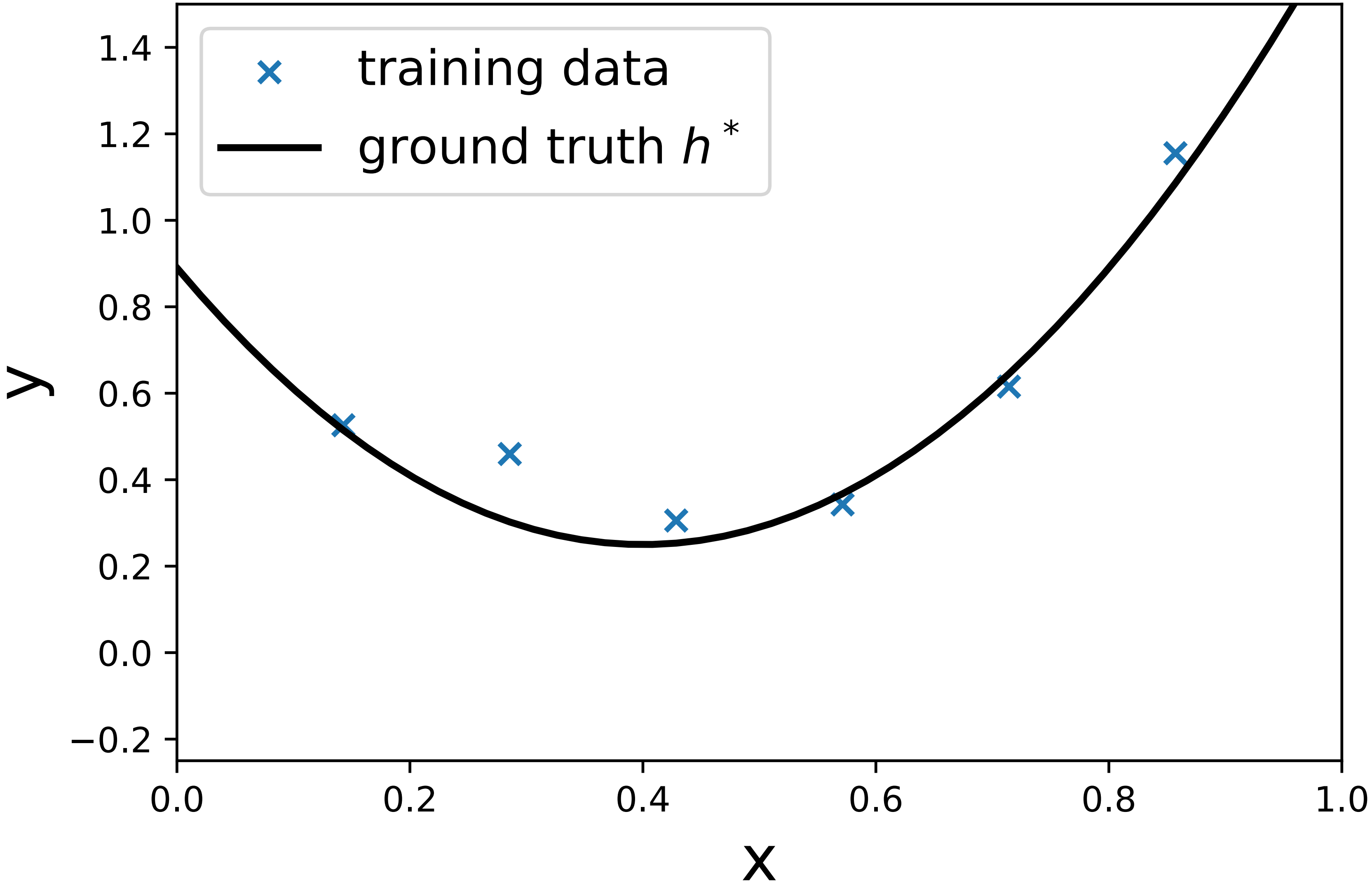
Training techniques

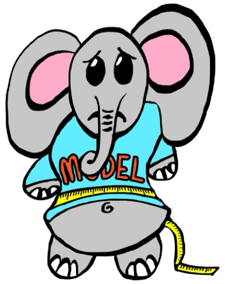
Statistical learning theory

Bias / variance trade-off

✋ Would love feedback + discussions!

Working example

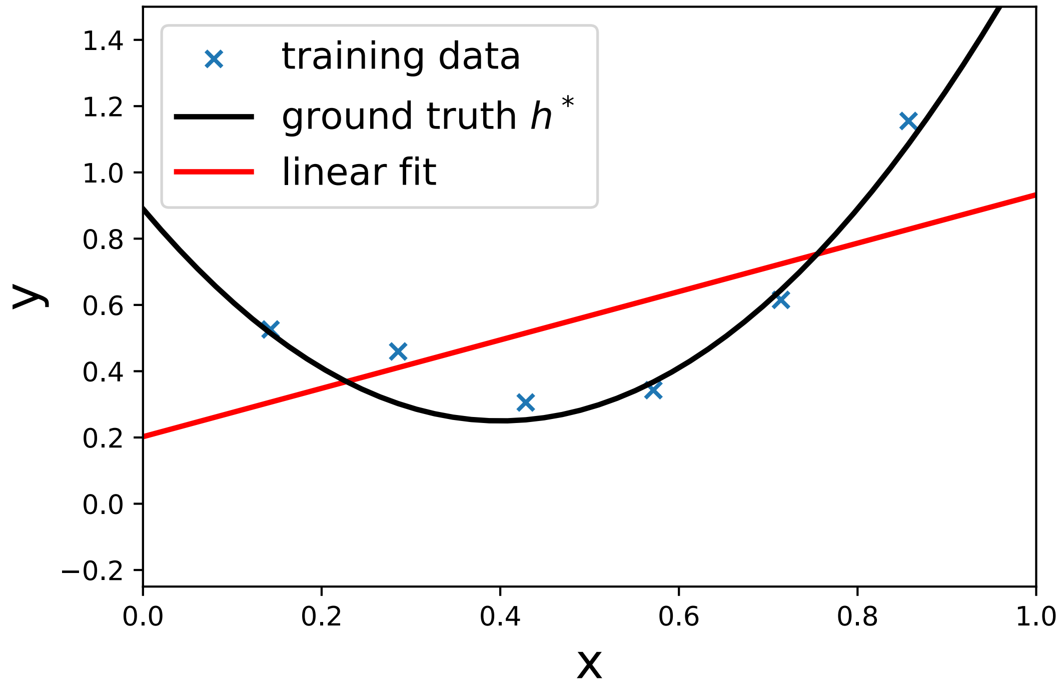




Underfitting

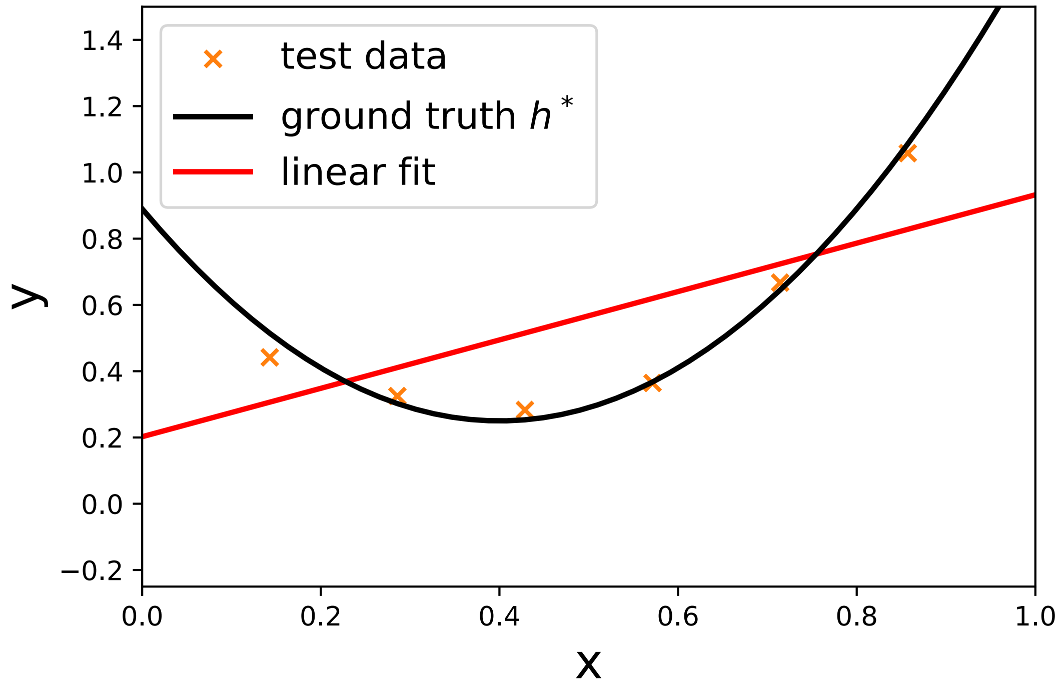
Model is not expressive enough

Training data

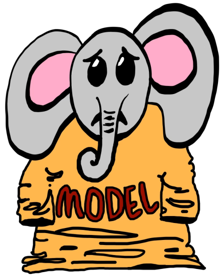


High training error

Test data



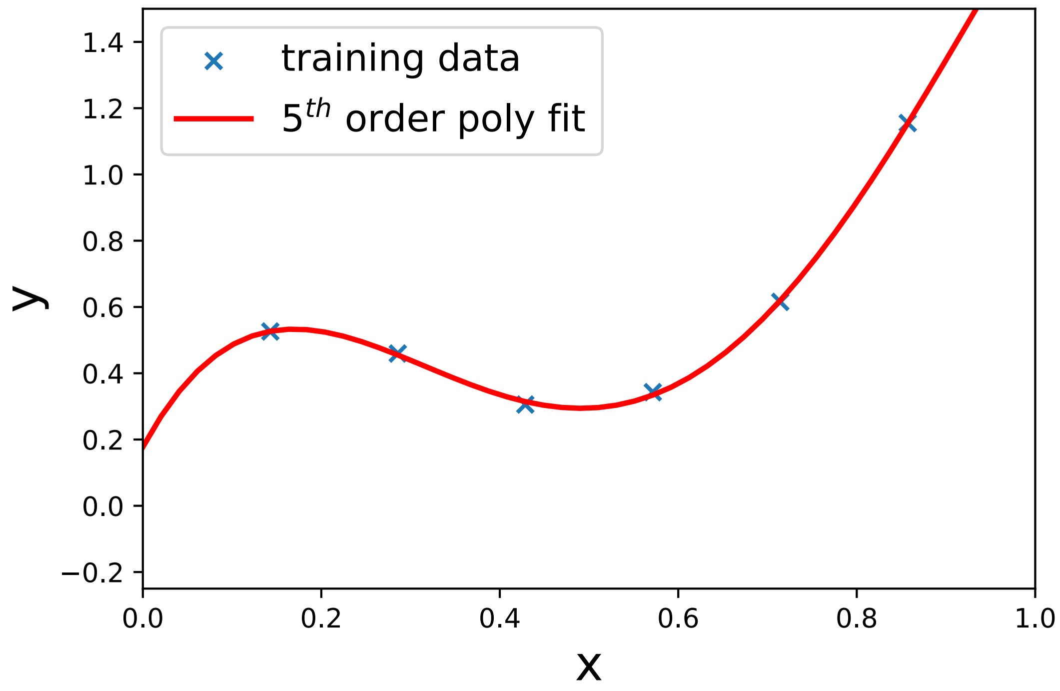
Also high test error



Overfitting

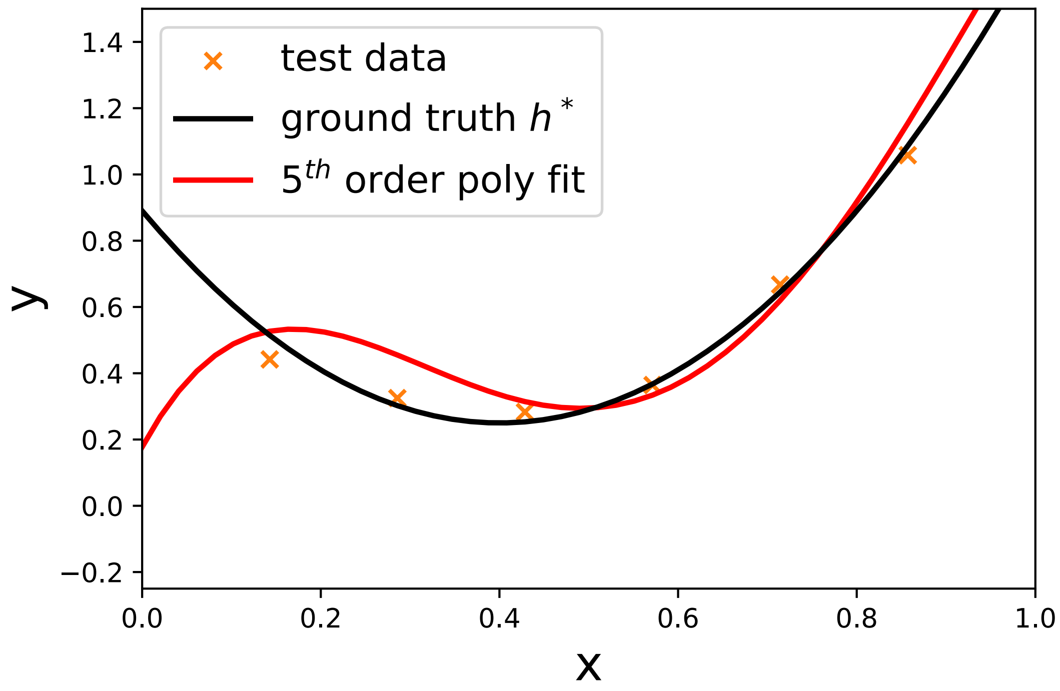
Model too expressive to generalize to unseen dataset

Training data



Small (zero) training error

Test data

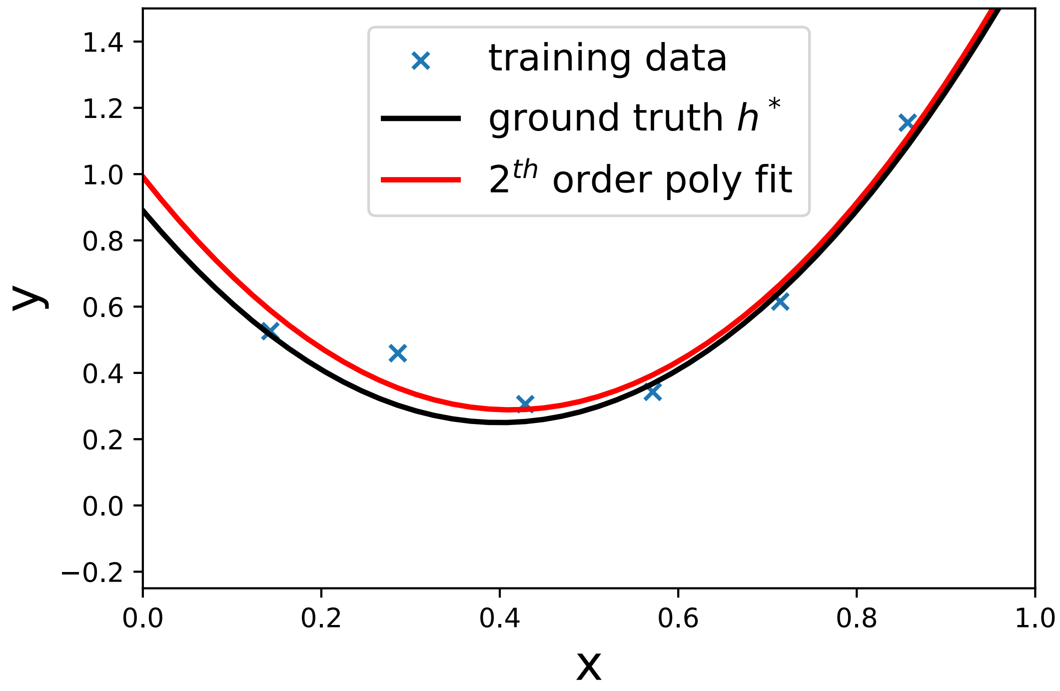


High test error

Optimal model complexity

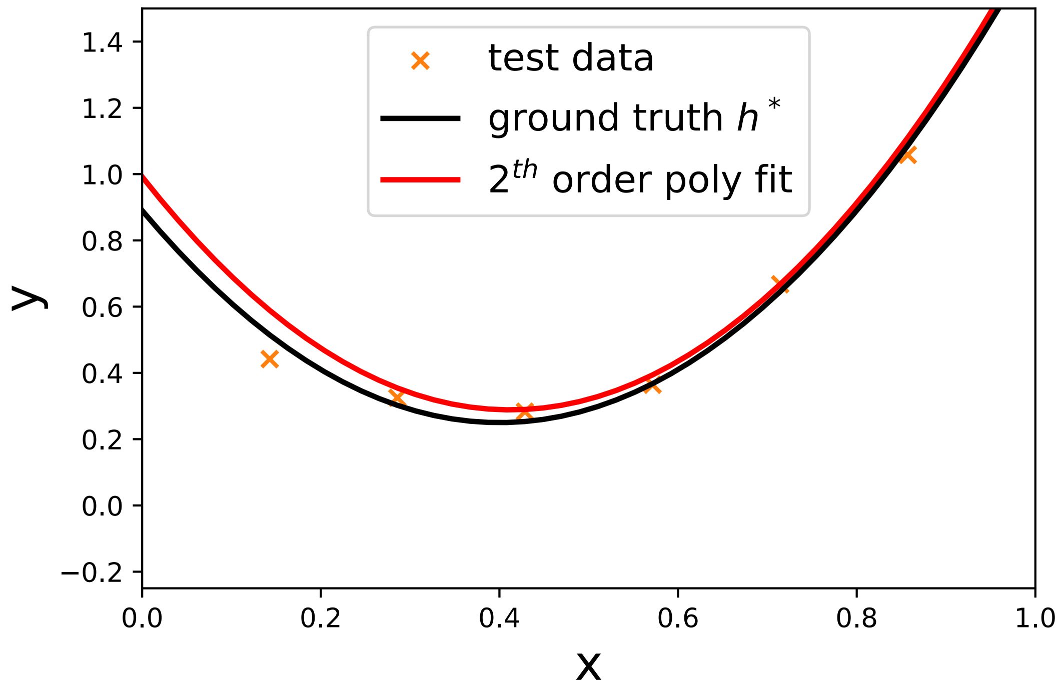
Fit 2nd order polynomial to quadratic distribution

Training data



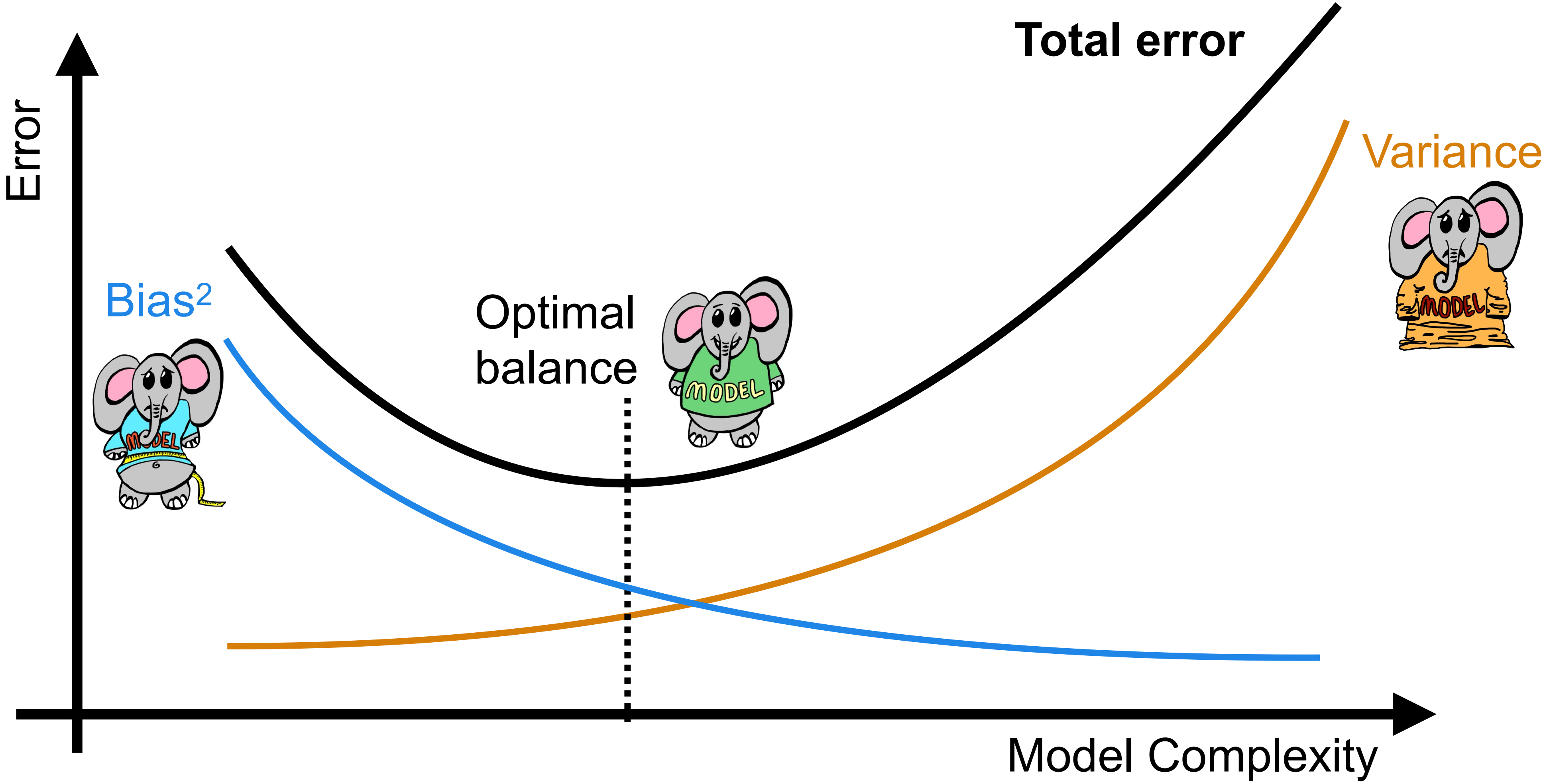
Small training error

Test data



Also small test error

Bias / variance tradeoff



Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0,\sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right]$$

Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0, \sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right] \overset{\text{Defn of } y}{=} \mathbb{E}\left[(h^*(x) + \xi - h_S(x))^2\right]$$

Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0,\sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right] \overset{\text{Defn of } y}{=} \mathbb{E}\left[(h^*(x) + \xi - h_S(x))^2\right]$$

$$= \mathbb{E}\left[(\xi \ + \ (h^*(x) - h_S(x)) \ )^2\right]$$

Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0, \sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right] \overset{\text{Defn of } y}{=} \mathbb{E}\left[(h^*(x) + \xi - h_S(x))^2\right]$$

$$= \mathbb{E}\left[(\xi \; + \; (h^*(x) - h_S(x)) \;)^2\right]$$

$$= \mathbb{E}\left[\xi^2\right] + 2\ \mathbb{E}[\xi] \cdot \mathbb{E}[h^*(x) - h_S(x)] + \mathbb{E}\left[(h^*(x) - h_S(x))^2\right]$$

Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0,\sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right] \overset{\text{Defn of } y}{=} \mathbb{E}\left[(h^*(x) + \xi - h_S(x))^2\right]$$

$$= \mathbb{E}\left[(\xi \;+\; (h^*(x) - h_S(x))\;)^2\right]$$

$$= \mathbb{E}\left[\xi^2\right] + 2\,\underbrace{\mathbb{E}[\xi]}_{0} \cdot \mathbb{E}[h^*(x) - h_S(x)] + \mathbb{E}\left[(h^*(x) - h_S(x))^2\right]$$

Bias / variance tradeoff: maths 1

- Training dataset $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$
  - Truth labels $y = h^*(x) + \xi$
    - h^* : ground truth function
    - $\xi^{(i)} \sim \mathcal{N}(0,\sigma^2)$
- Train model h_S on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

$$MSE(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right] \overset{\text{Defn of } y}{=} \mathbb{E}\left[(h^*(x) + \xi - h_S(x))^2\right]$$

$$= \mathbb{E}\left[(\xi \;+\; (h^*(x) - h_S(x)) \;)^2\right]$$

$$= \mathbb{E}\left[\xi^2\right] + 2\,\underbrace{\mathbb{E}[\xi]}_{0} \cdot \mathbb{E}[h^*(x) - h_S(x)] + \mathbb{E}\left[(h^*(x) - h_S(x))^2\right]$$

$$= \sigma^2 + \mathbb{E}\left[(h^*(x) - h_S(x))^2\right]$$

Bias / variance tradeoff: maths 2

- Let $h_{avg}(x) = \mathbb{E}_S\left[h_S(x)\right]$ – the performance of the model trained on infinitely many datasets

$$MSE(x) = \sigma^2 + \mathbb{E}\left[(h^*(x) - h_S(x))^2\right]$$

$$= \sigma^2 + \mathbb{E}\left[\ (\ h^*(x) - h_{avg}(x) + h_{avg}(x) - h_S(x)\ )^2\ \right]$$

$$= \sigma^2 + \mathbb{E}\left[(h^*(x) - h_{avg}(x))^2\ +\ ((h_{avg}(x) - h_S(x))^2\right]$$

No cross-term because $\mathbb{E}\left[h_{avg}(x) - h_S(x)\right] = 0$

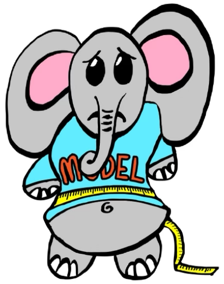
$$= \sigma^2 + (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

Bias^2

Error on this class of models

Variance

How does this instantiation compare with the other possible ones?



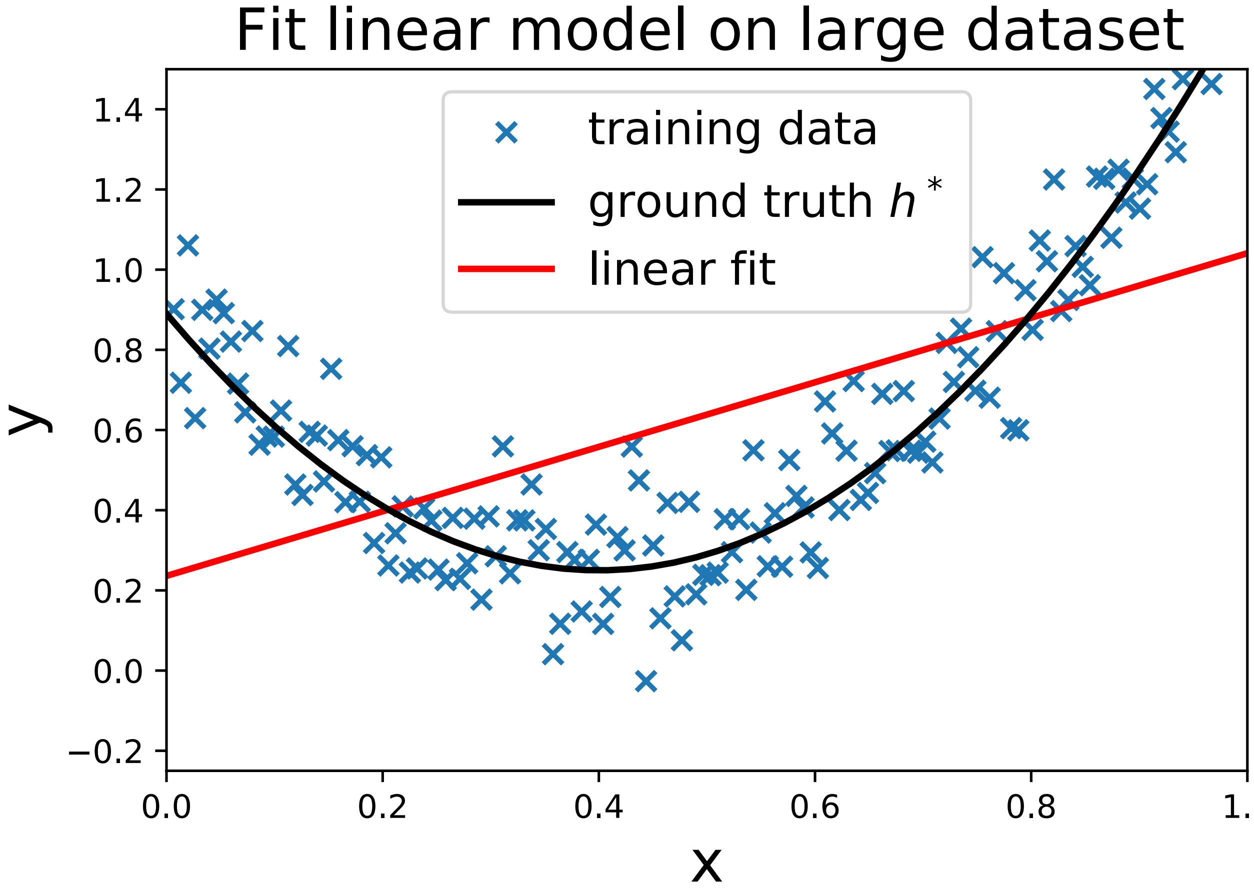
High bias: diagnostics

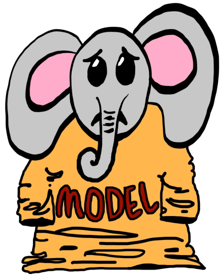
$$MSE(x) = \underbrace{(h^*(x) - h_{avg}(x))^2}_{\text{Bias}^2} + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

Linear fit

$$h_S(x) = \theta_0 + \theta_1 x$$

The training error high, even if we increase the training data.





High variance: diagnostics

$$MSE(x) = (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

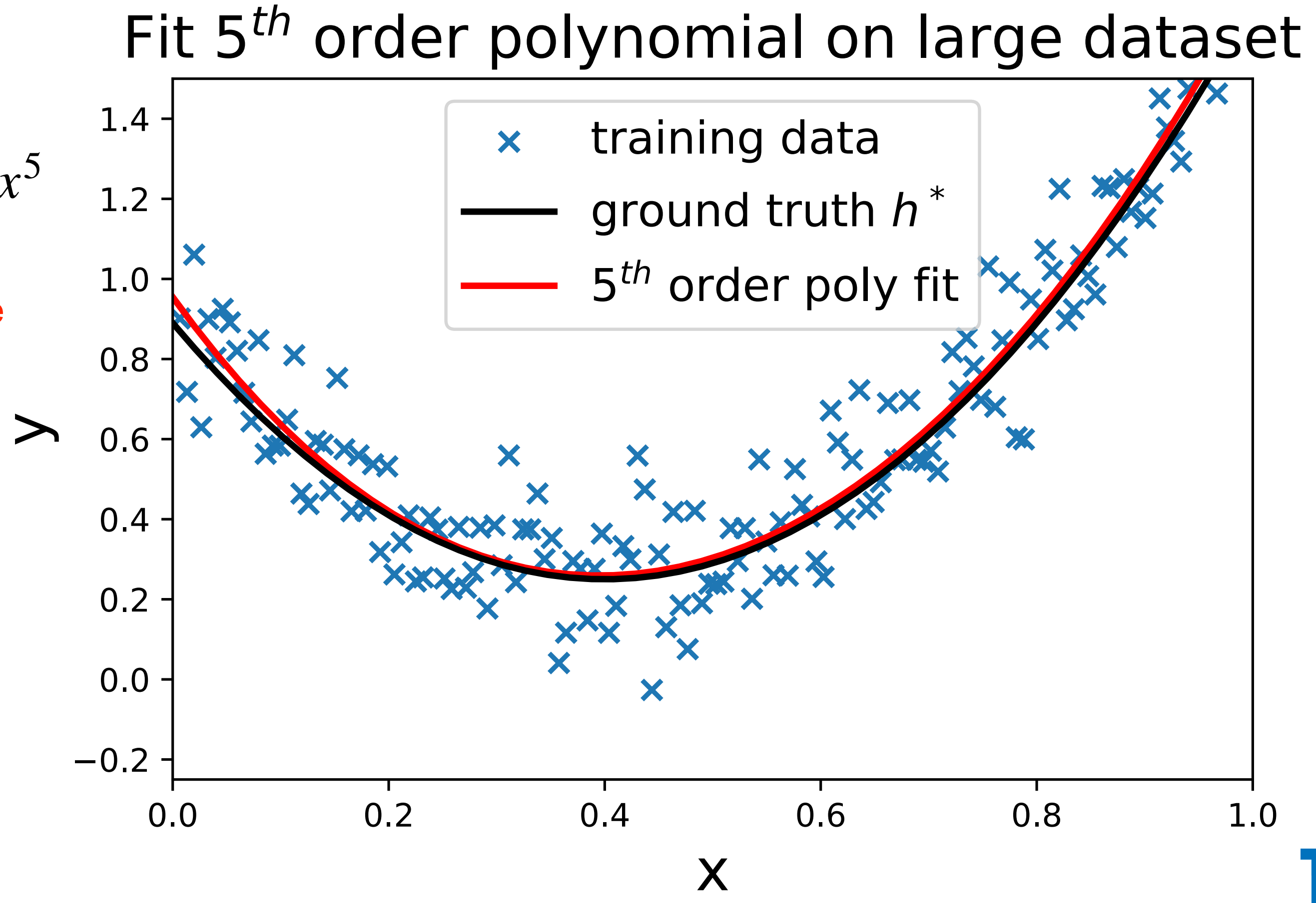
Variance

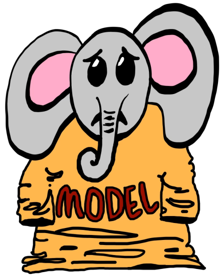
<u>5th order polynomial fit</u>

$$h_S(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

Can learn to set these coefficients to 0 (for θ_3, θ_4, θ_5)

The training error decreases as we increase the training data.

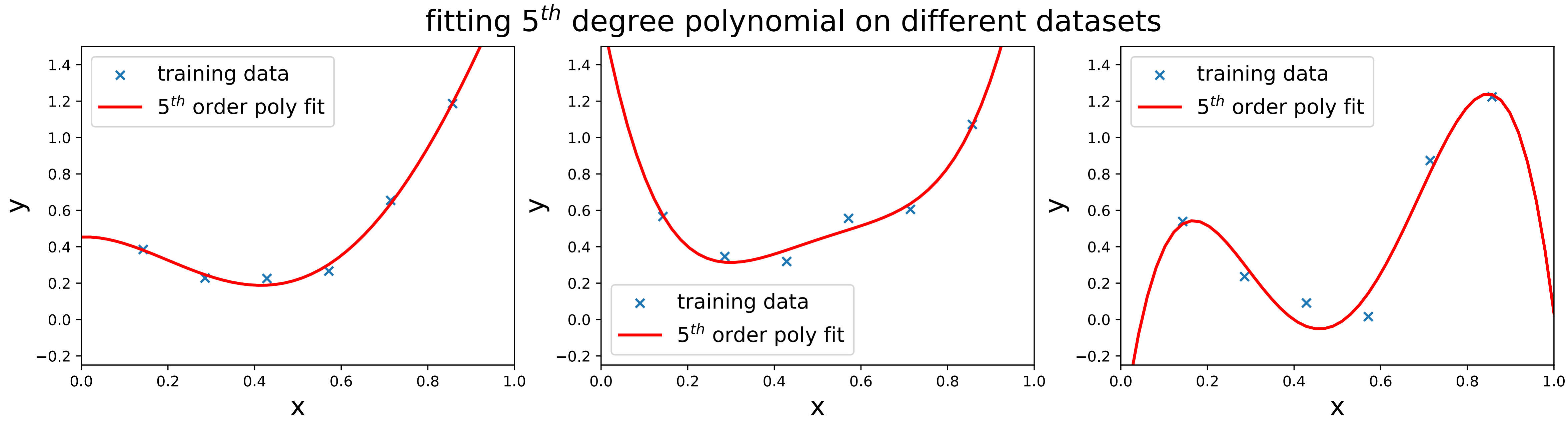




High variance: intuition

$$MSE(x) = (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

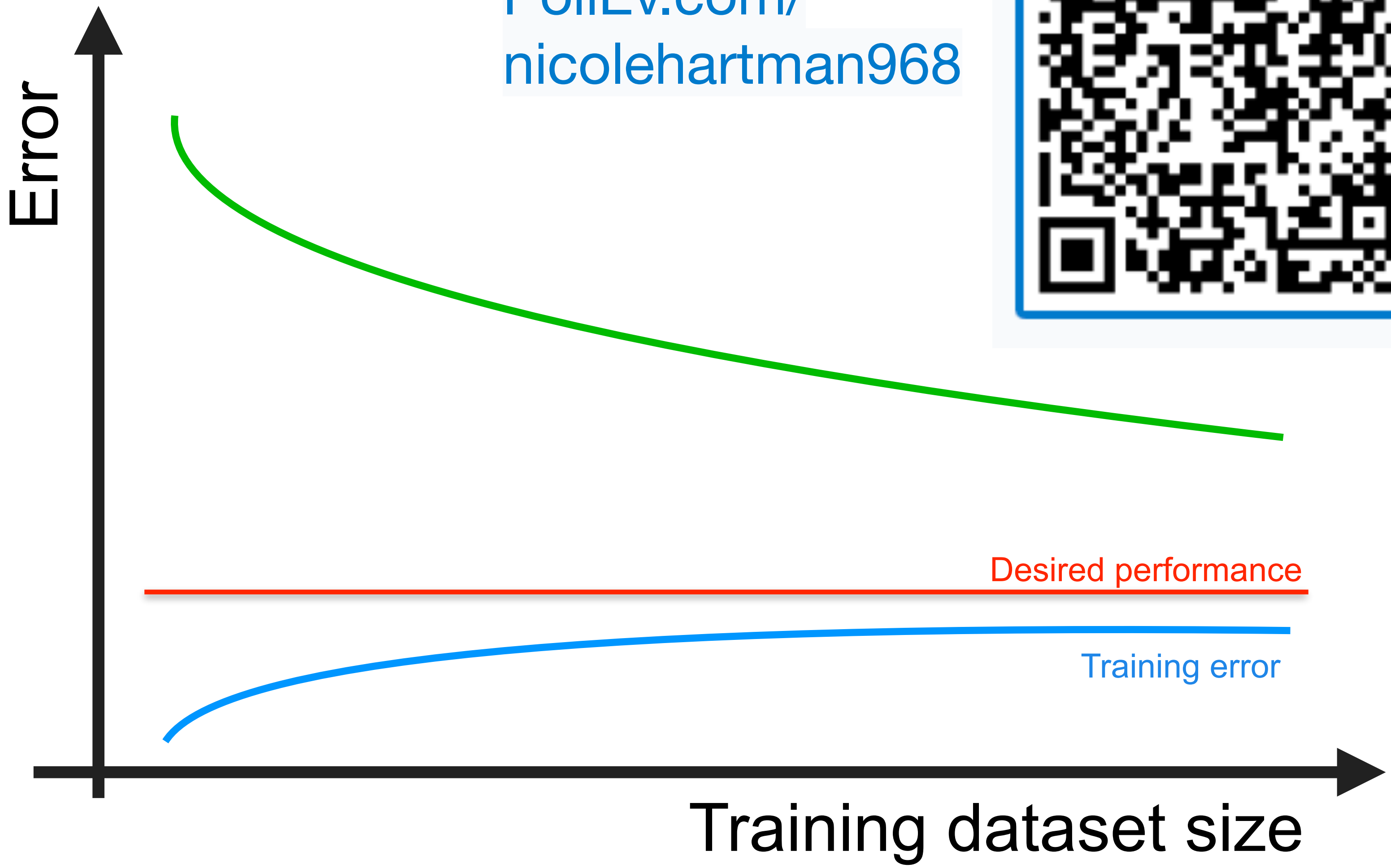
Variance



Lots of possibilities for the fitted function depending on the random realization of training data.

What's the culprit?

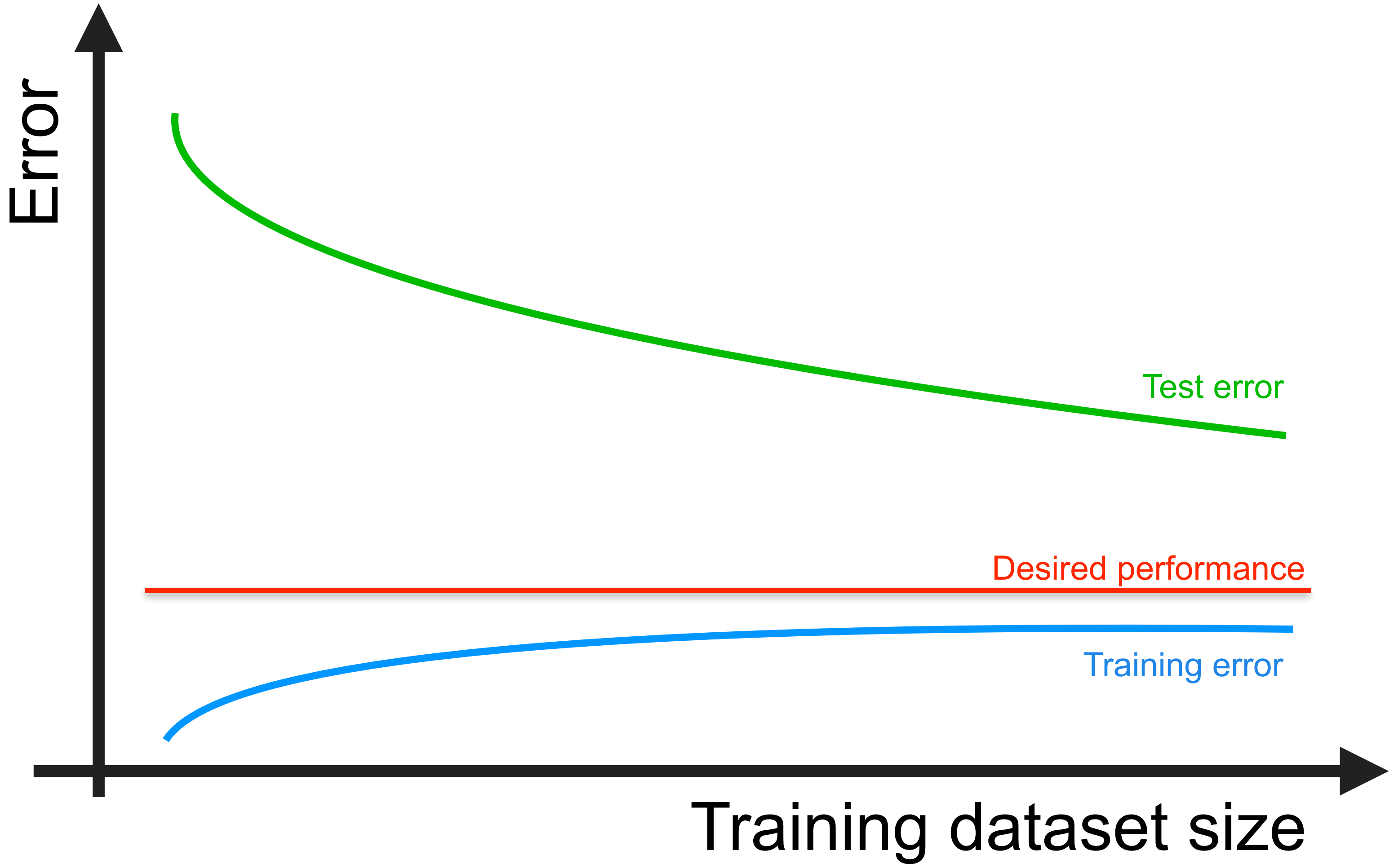
PollEv.com/
nicolehartman968



High bias?

High Variance?

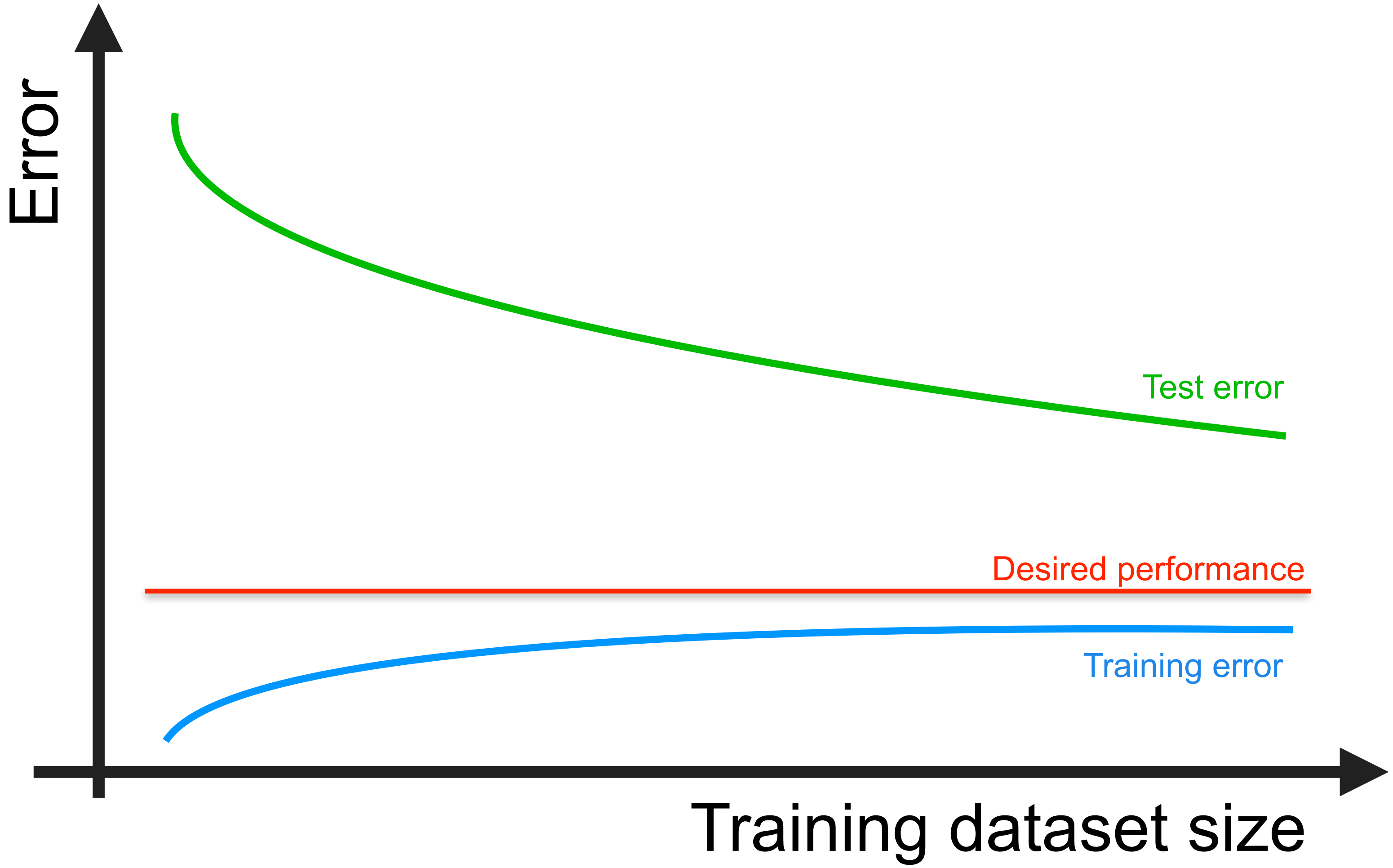
What’s the culprit?



High bias?

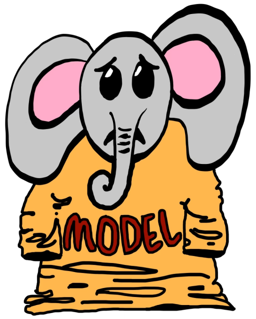
High Variance?

What’s the culprit?

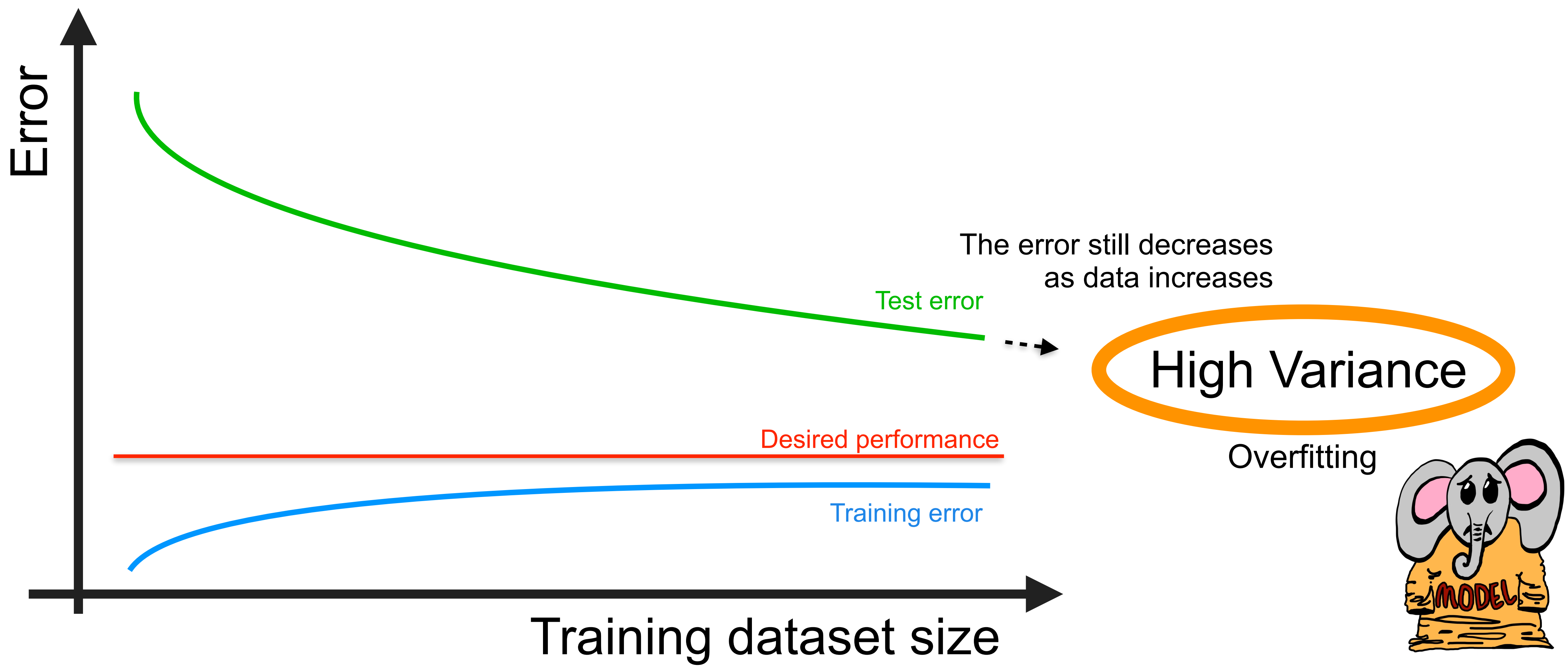


High Variance

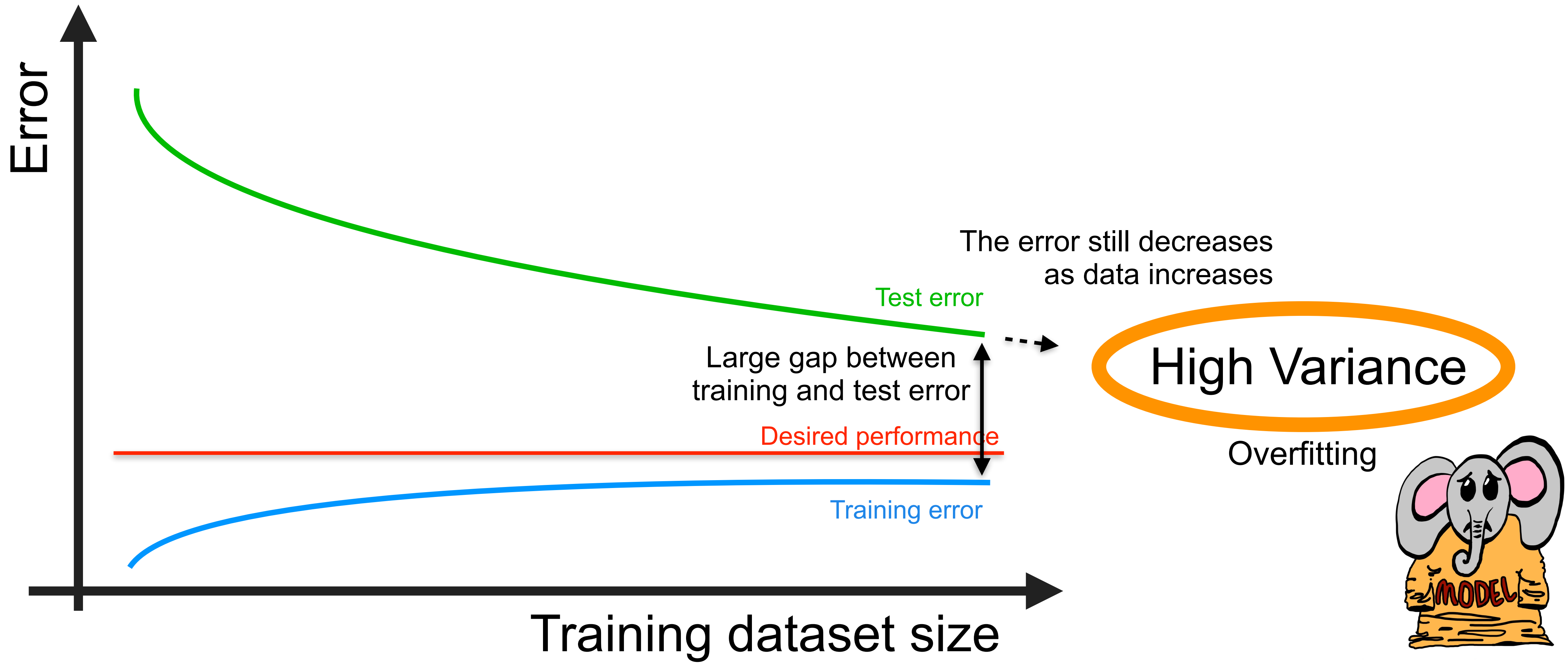
Overfitting



What's the culprit?

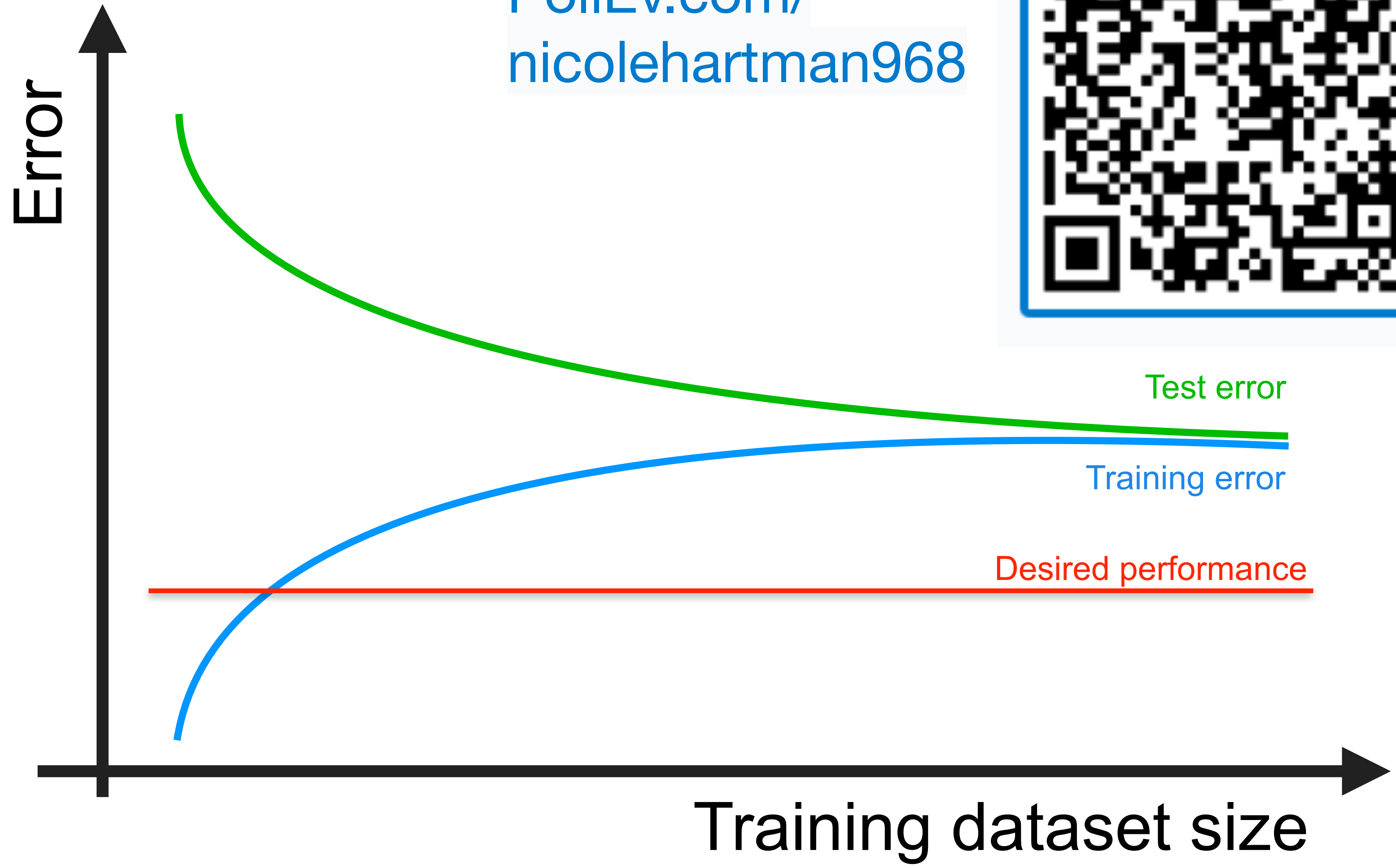


What's the culprit?



What's the culprit?

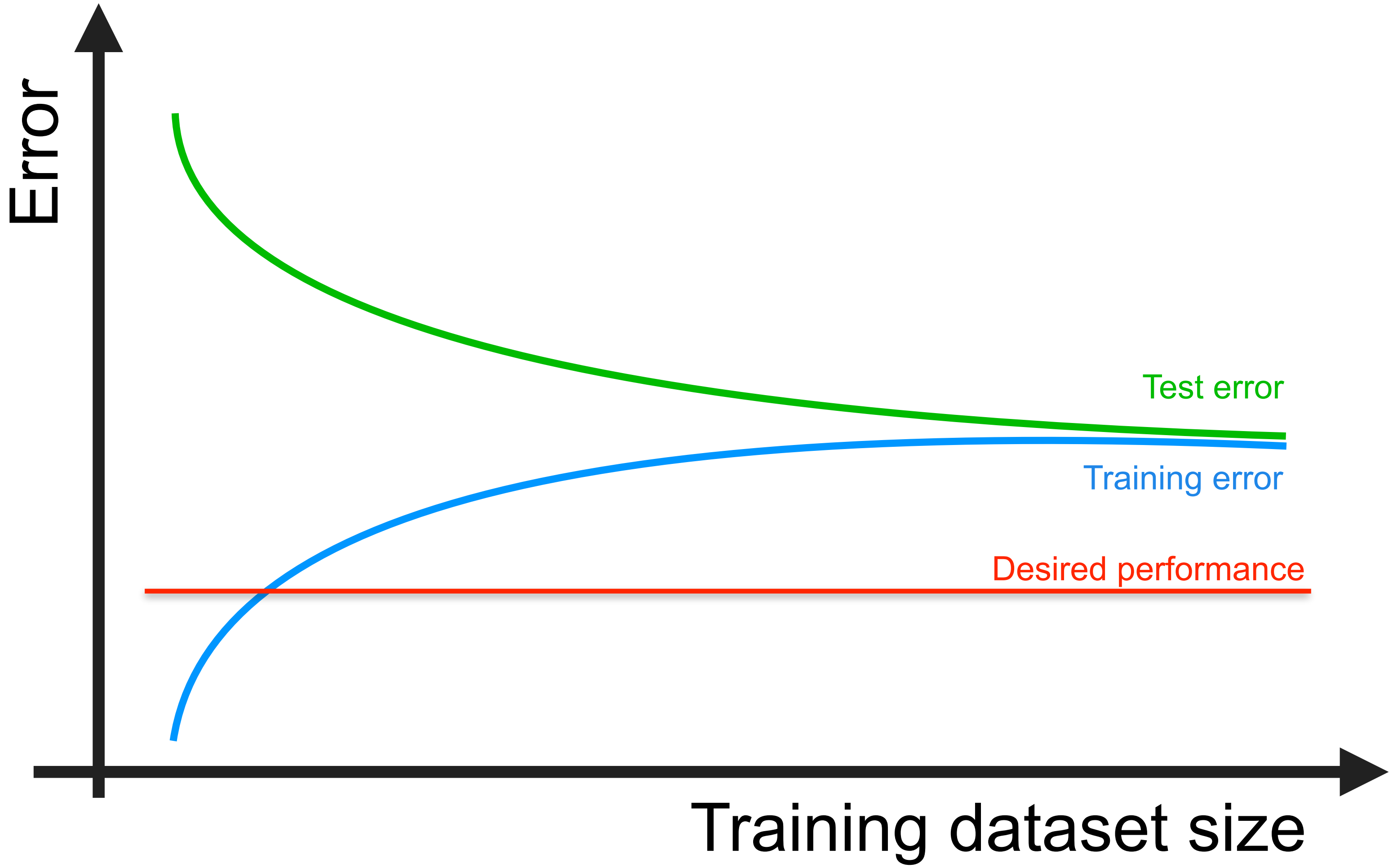
PollEv.com/
nicolehartman968



High bias?

High Variance?

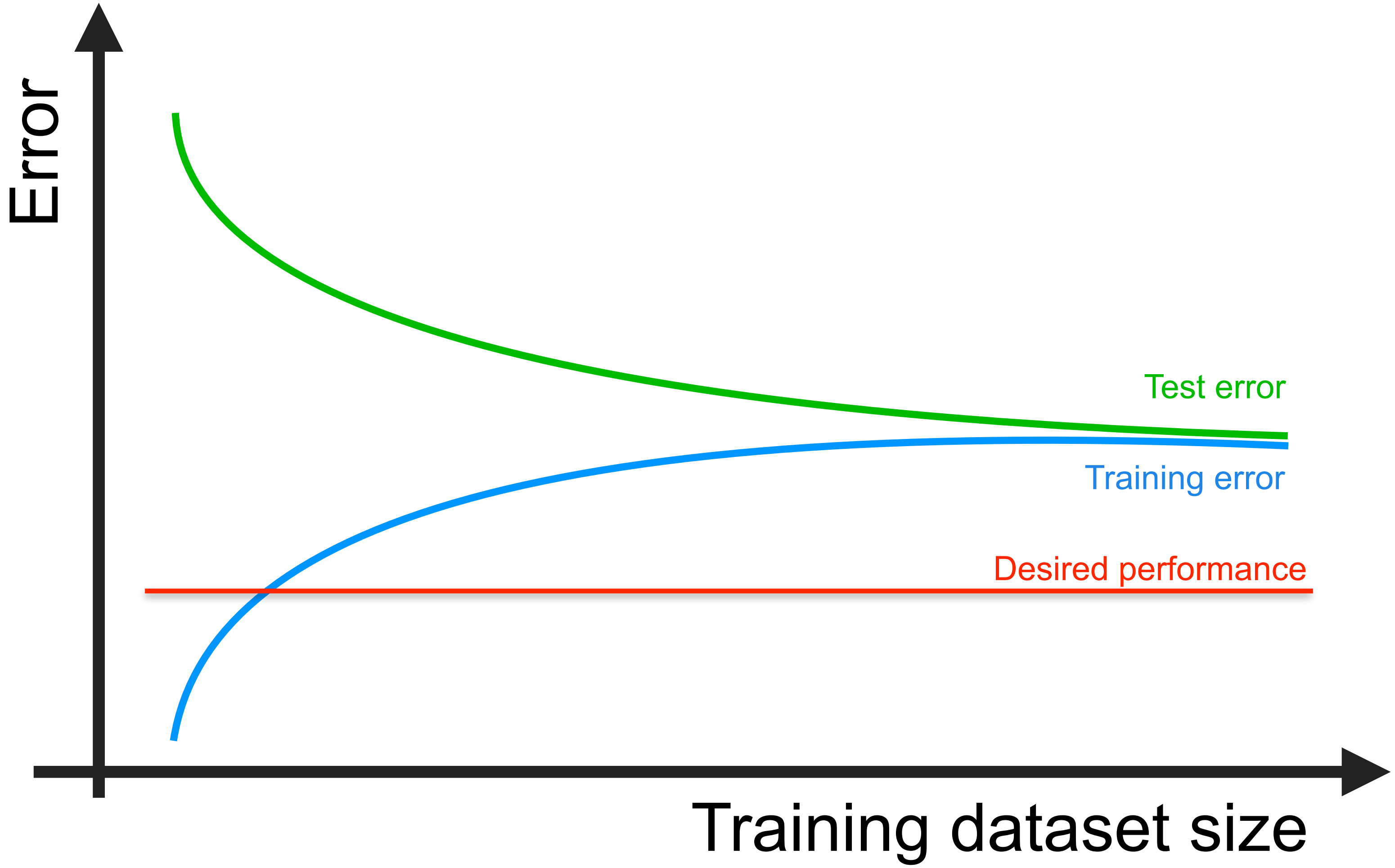
What's the culprit?



High bias?

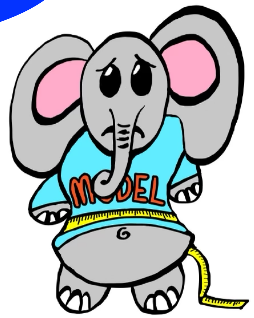
High Variance?

What's the culprit?

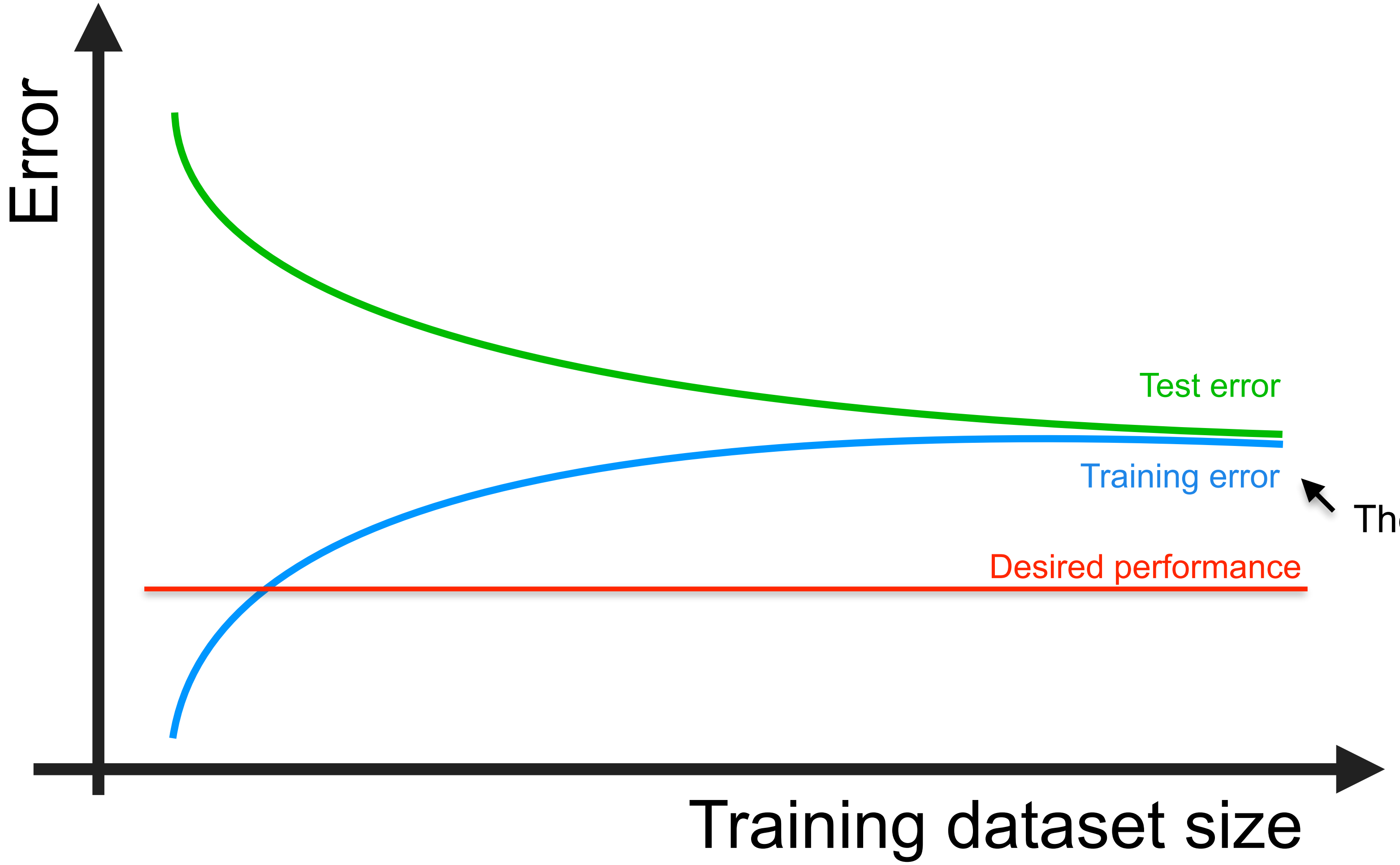


High bias

Underfitting



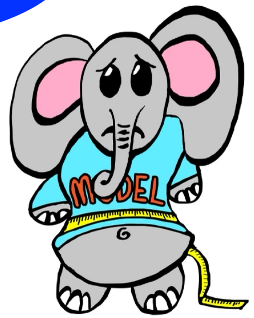
What's the culprit?



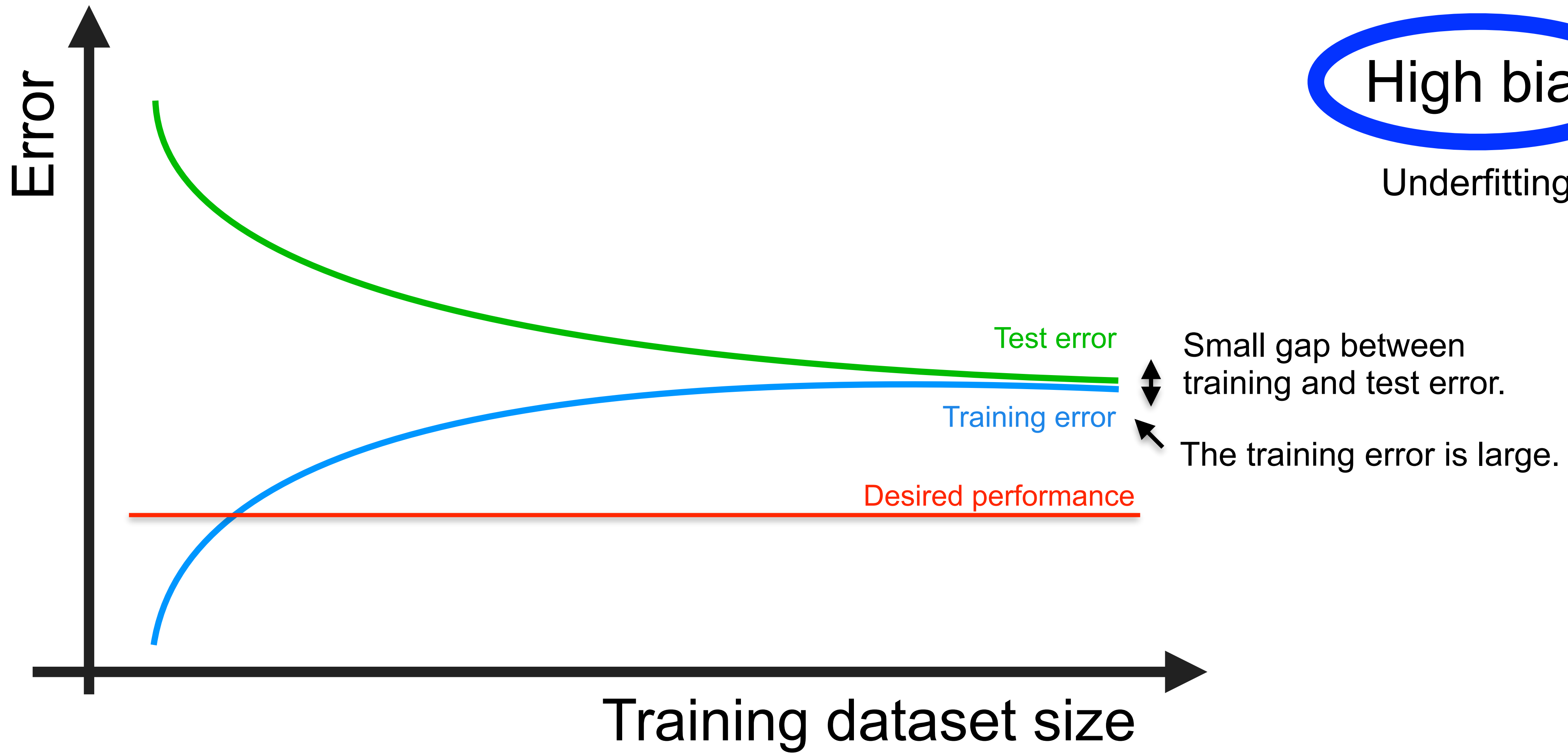
The training error is large.

High bias

Underfitting

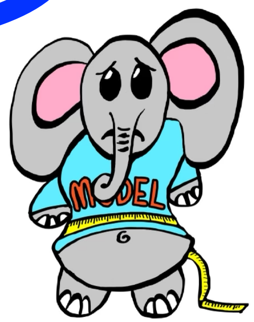


What's the culprit?



High bias

Underfitting



Starting off...

Open question

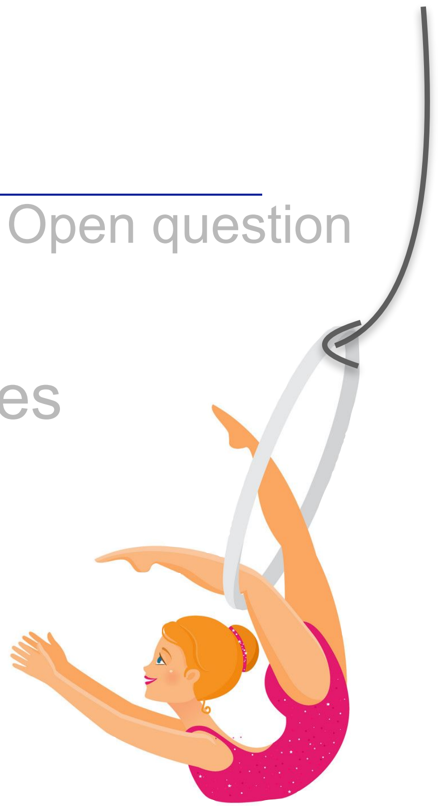
Feature choices

Going deeper

Training techniques

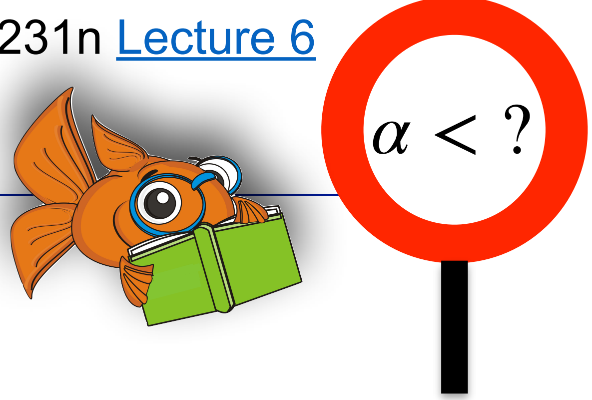
Statistical learning theory

Bias / variance trade-off



✋ Would love feedback + discussions!

CS231n Lecture 6

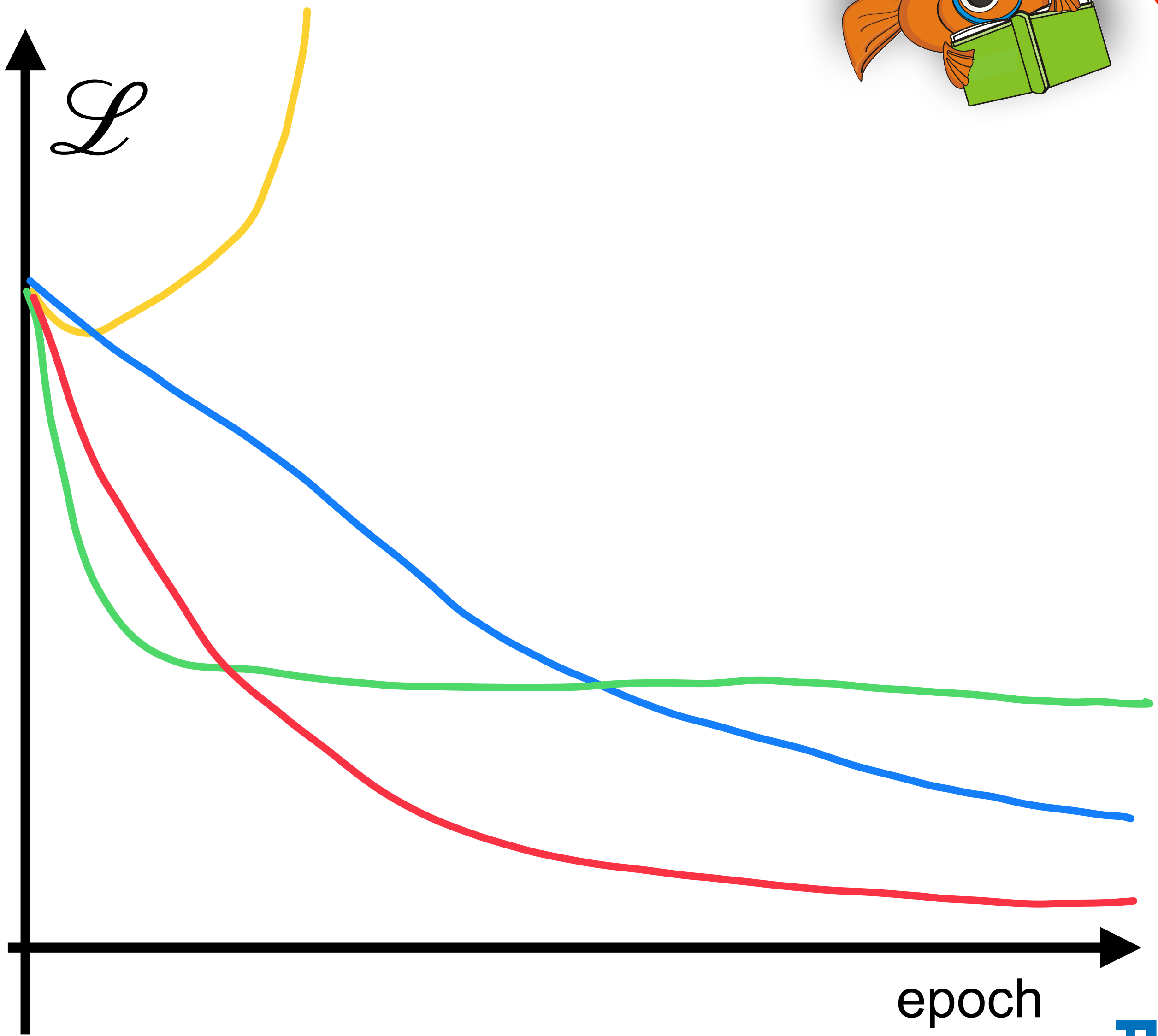


Learning rate

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

How to choose α?



Label the loss curves!

very high learning rate
high learning rate
good learning rate
low learning rate

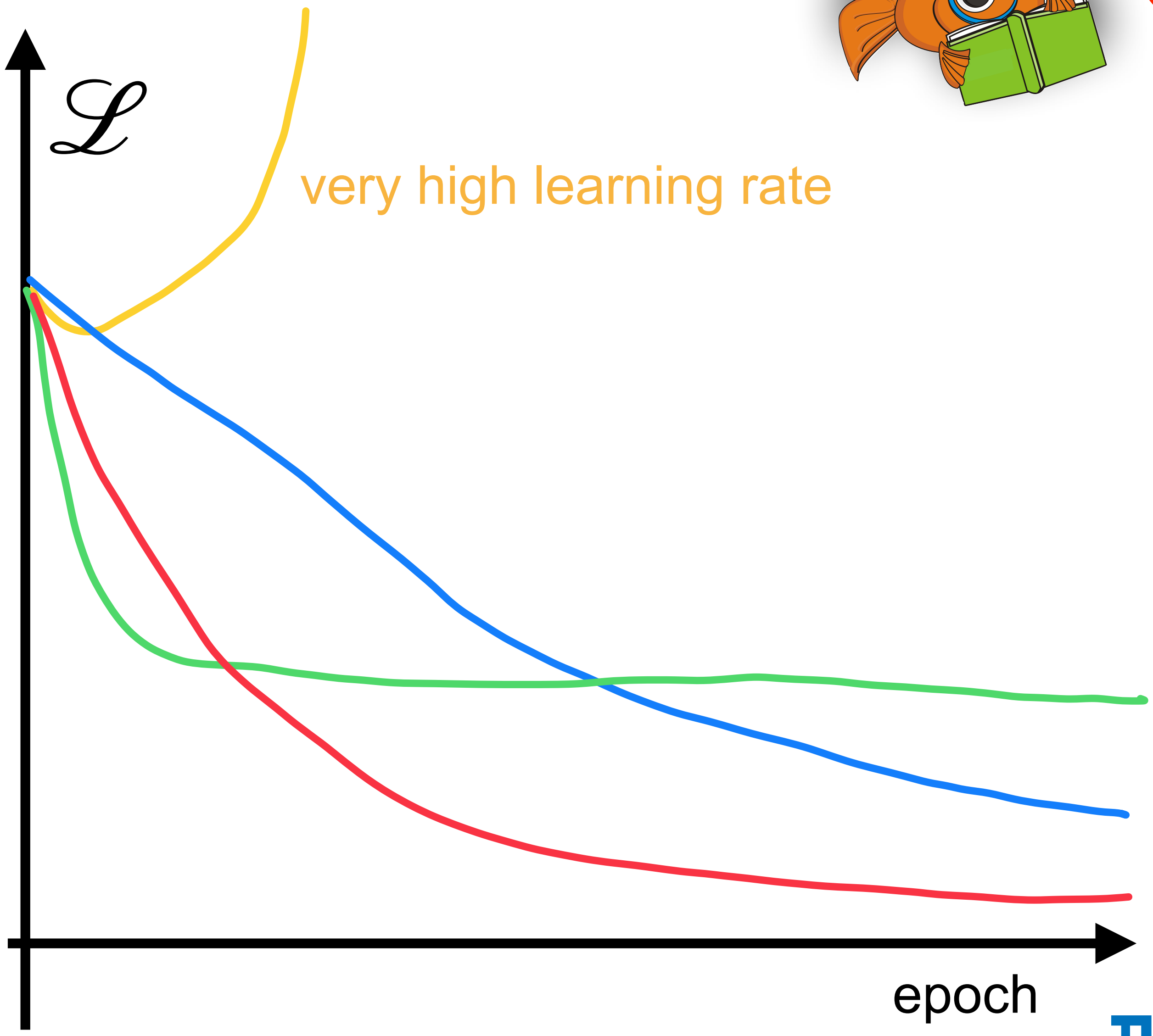
Learning rate

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

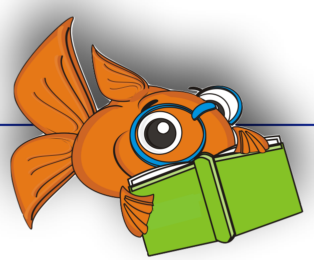
How to choose α?

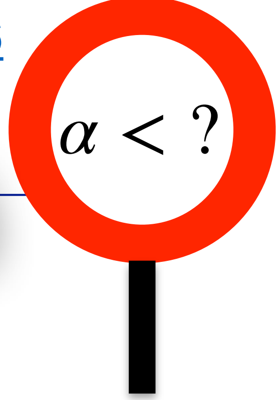
$\alpha < ?$



Label the loss curves!

very high learning rate
high learning rate
good learning rate
low learning rate





Learning rate

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

How to choose α?

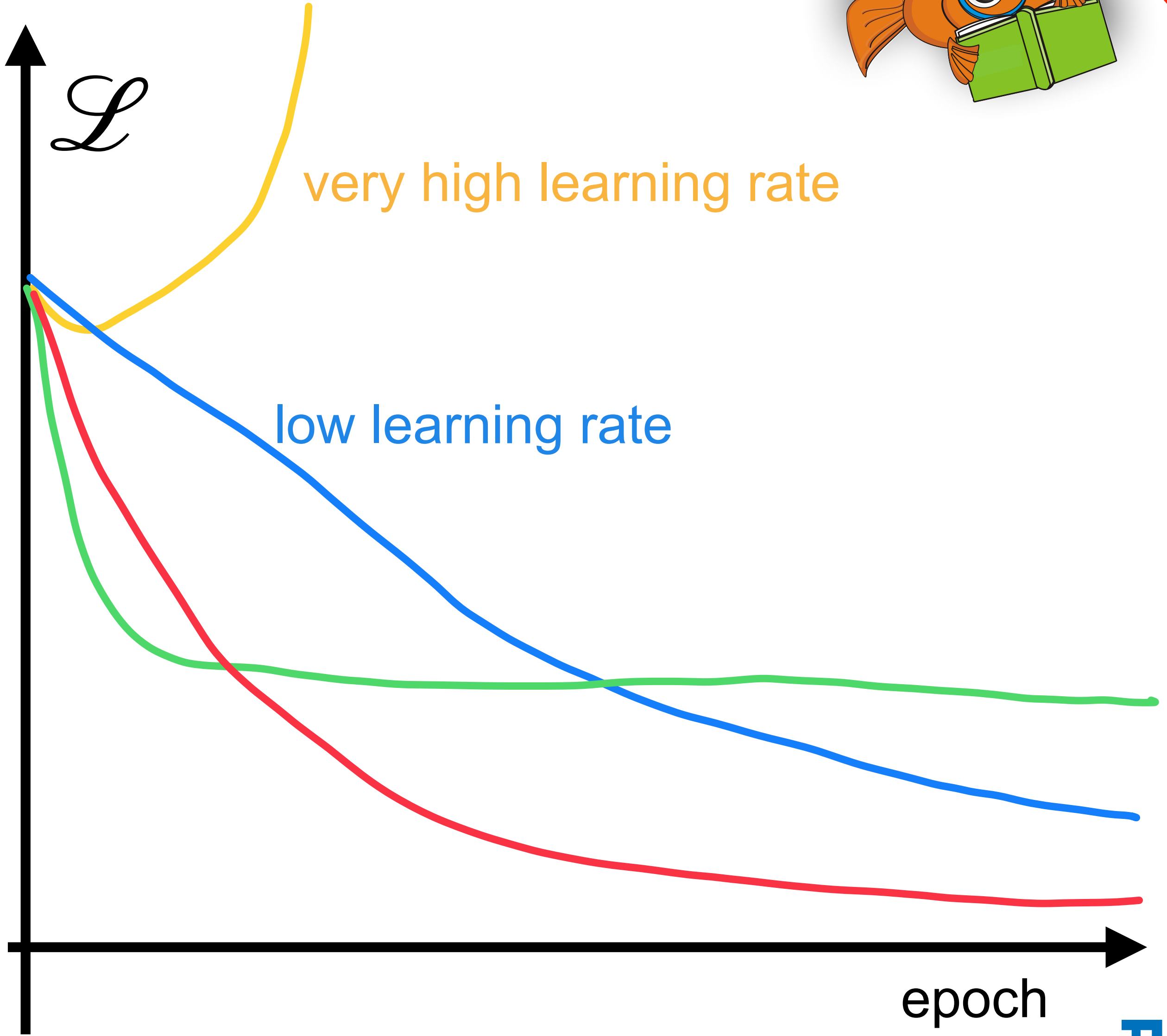
📝 Label the loss curves!

very high learning rate

high learning rate

good learning rate

low learning rate



Learning rate

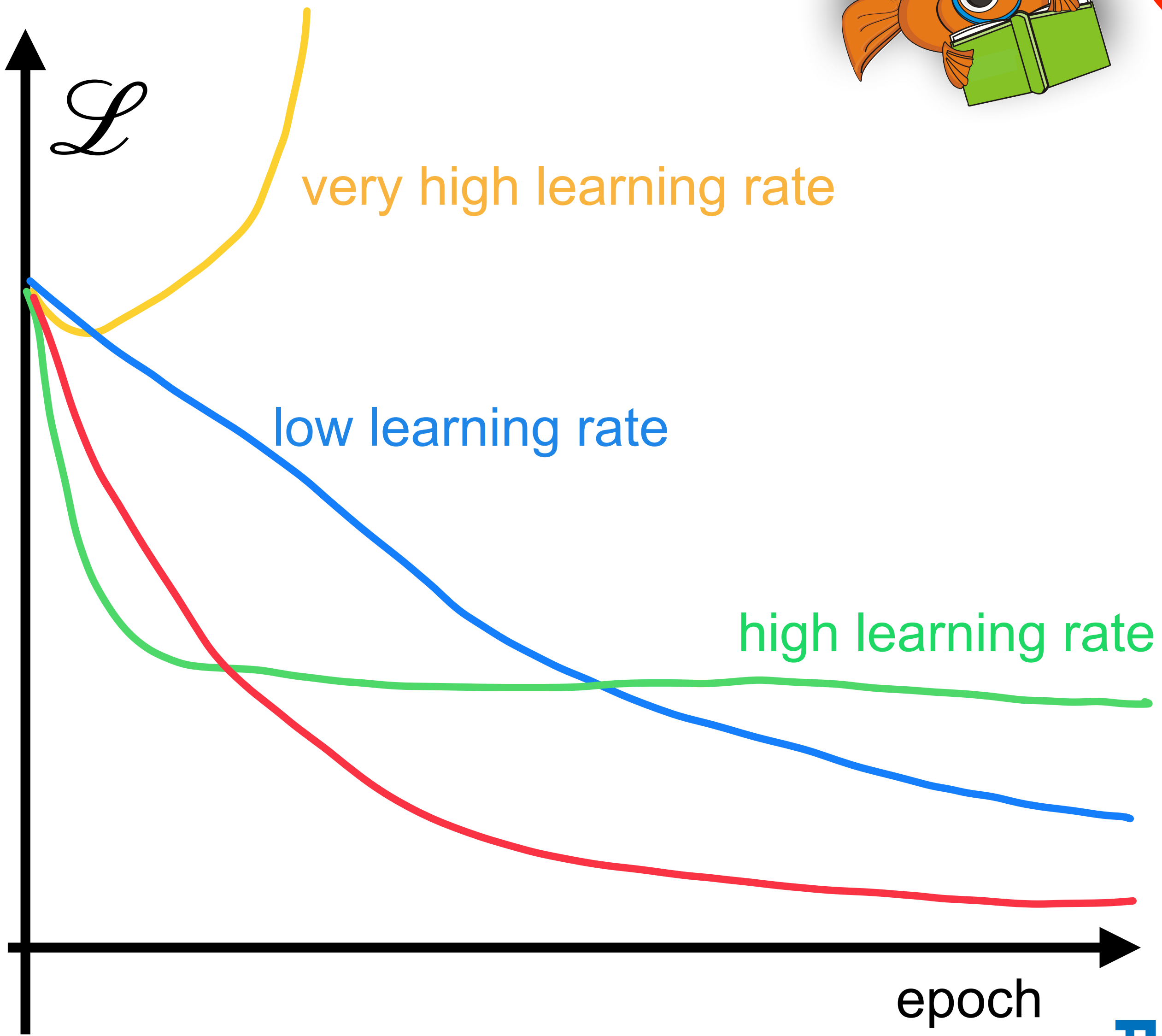
Minimize $\mathscr{L}$ by SGD

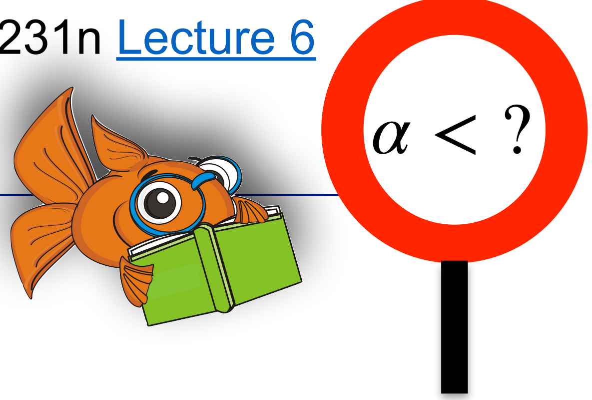
$$w = w - \alpha \nabla_w \mathscr{L}$$

How to choose α?

Label the loss curves!

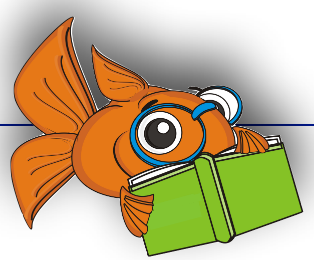
very high learning rate
high learning rate
good learning rate
low learning rate

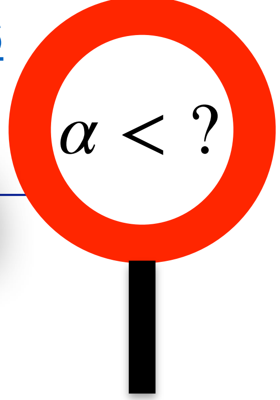




CS231n Lecture 6

Learning rate





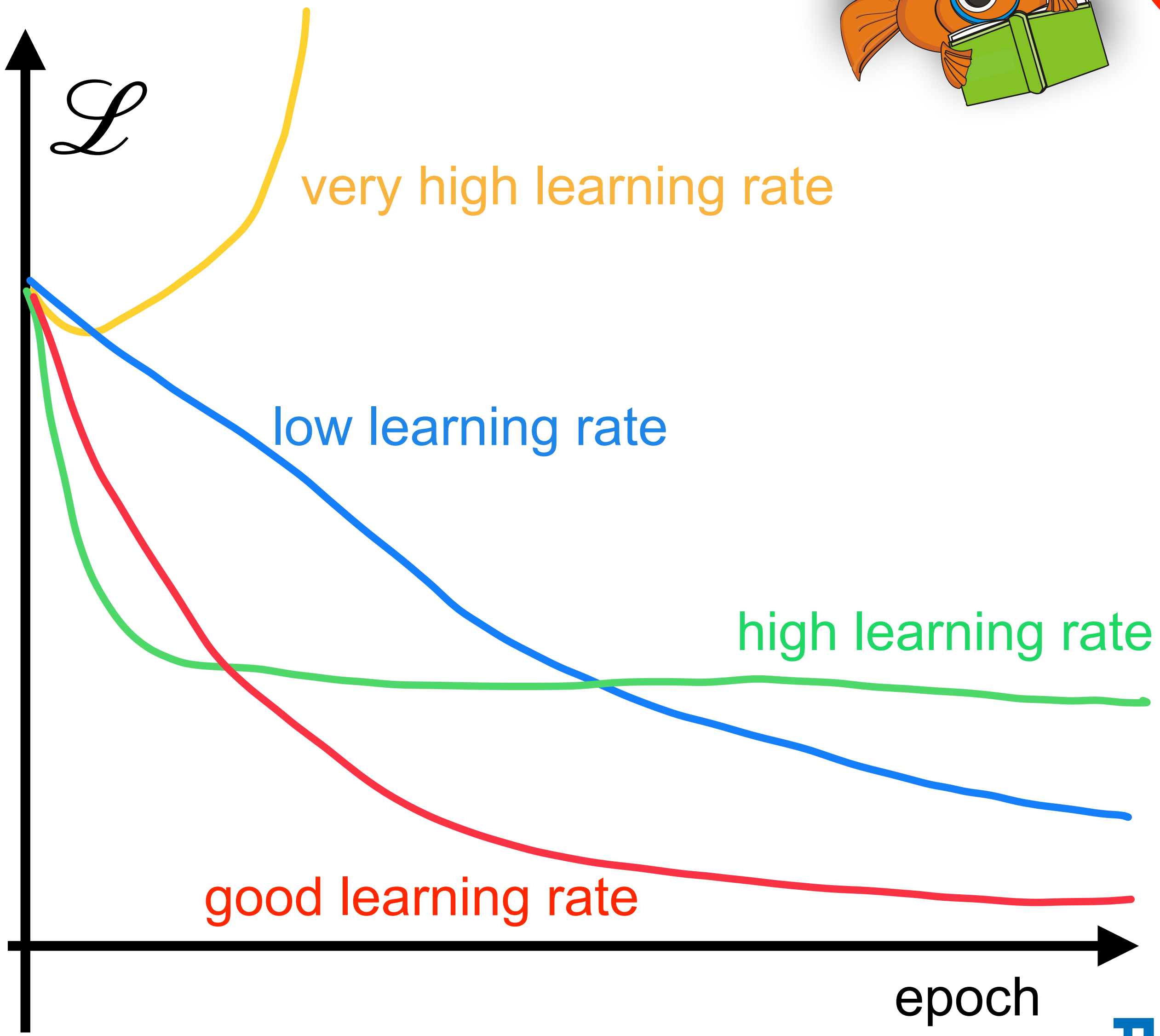
Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

How to choose α?

Label the loss curves!

very high learning rate
high learning rate
good learning rate
low learning rate



Batch size

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \boxed{\nabla_w \mathscr{L}}$$

Make a MC estimate

post

Batch size

Minimize $\mathscr{L}$ by SGD

m: mini-batch size

$$w = w - \alpha \boxed{\nabla_w \mathscr{L}} \approx w - \alpha \sum_{i=1}^{m} \nabla_w \mathscr{L}(x_i, y_i)$$

Make a MC estimate

post

Batch size

Minimize $\mathscr{L}$ by SGD

m: mini-batch size

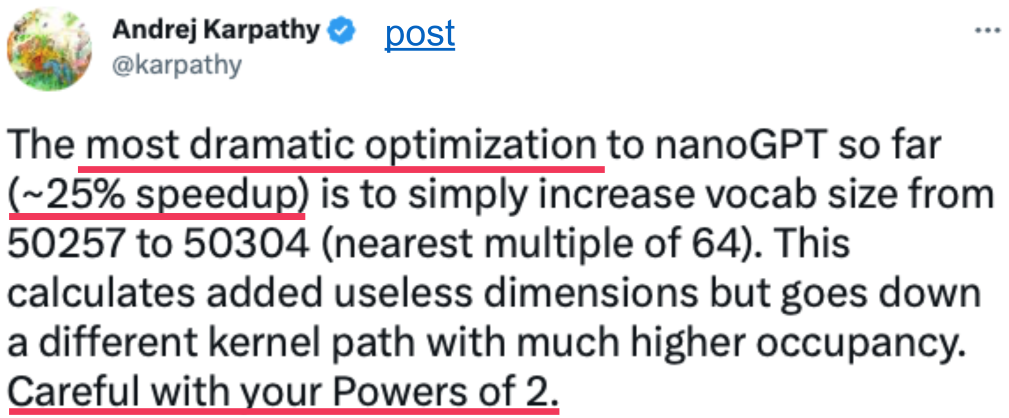
$$w = w - \alpha \boxed{\nabla_w \mathscr{L}} \approx w - \alpha \sum_{i=1}^{m} \nabla_w \mathscr{L}(x_i, y_i)$$

Make a MC estimate

Bigger batches reduces the error on the MC estimate

- As large as possible to still fit on the GPU.
- Powers of 2 for memory efficiency

  E.g, 256, 512, 1024


Andrej Karpathy post
@karpathy

The most dramatic optimization to nanoGPT so far (~25% speedup) is to simply increase vocab size from 50257 to 50304 (nearest multiple of 64). This calculates added useless dimensions but goes down a different kernel path with much higher occupancy. Careful with your Powers of 2.

Batch size

Minimize $\mathscr{L}$ by SGD

m: mini-batch size

$$w = w - \alpha \boxed{\nabla_w \mathscr{L}} \approx w - \alpha \sum_{i=1}^{m} \nabla_w \mathscr{L}(x_i, y_i)$$

Make a MC estimate

Bigger batches reduces the error on the MC estimate

- As large as possible to still fit on the GPU.
- Powers of 2 for memory efficiency

  E.g, 256, 512, 1024

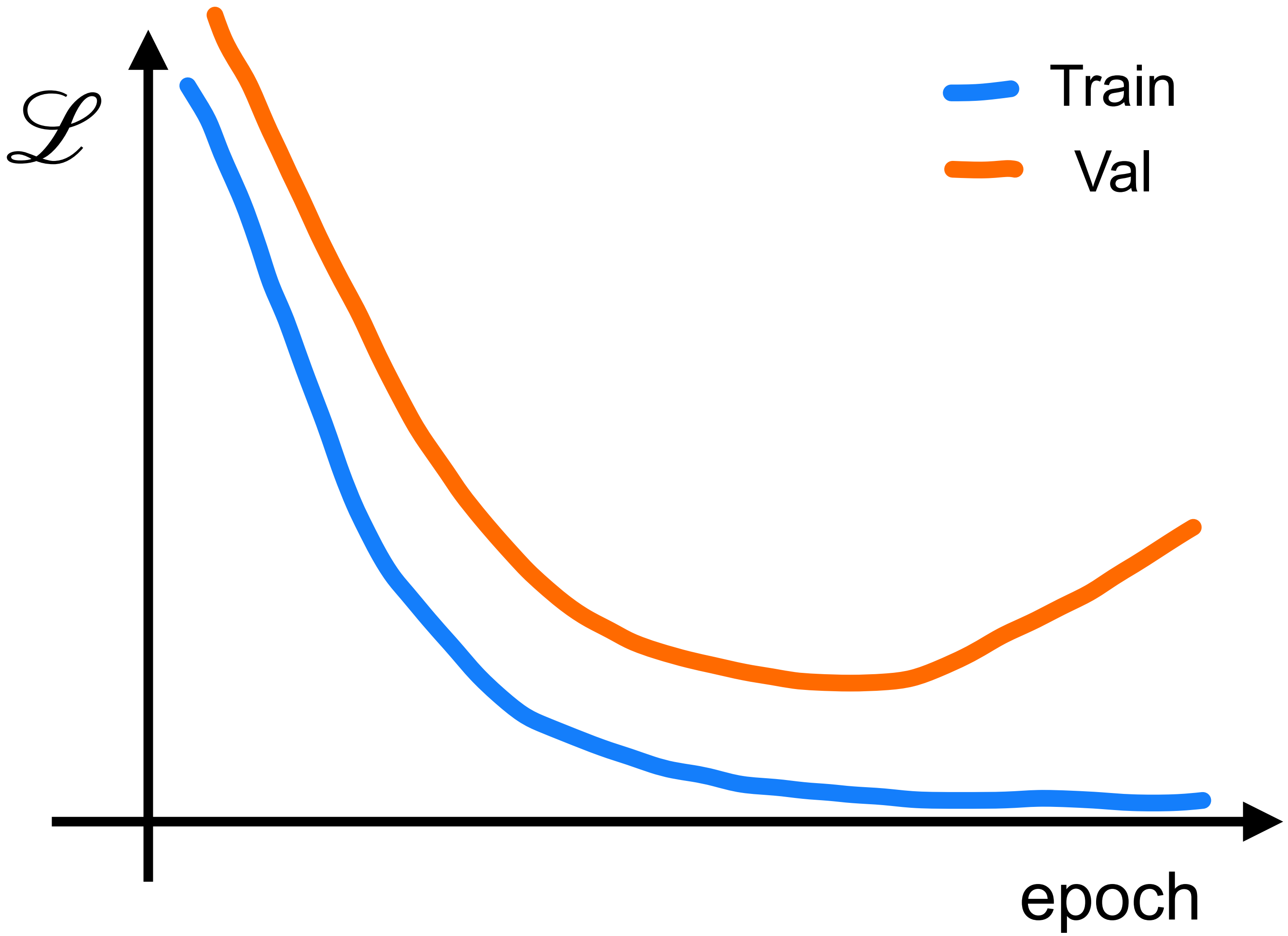
Andrej Karpathy @karpathy post

The most dramatic optimization to nanoGPT so far (~25% speedup) is to simply increase vocab size from 50257 to 50304 (nearest multiple of 64). This calculates added useless dimensions but goes down a different kernel path with much higher occupancy. Careful with your Powers of 2.

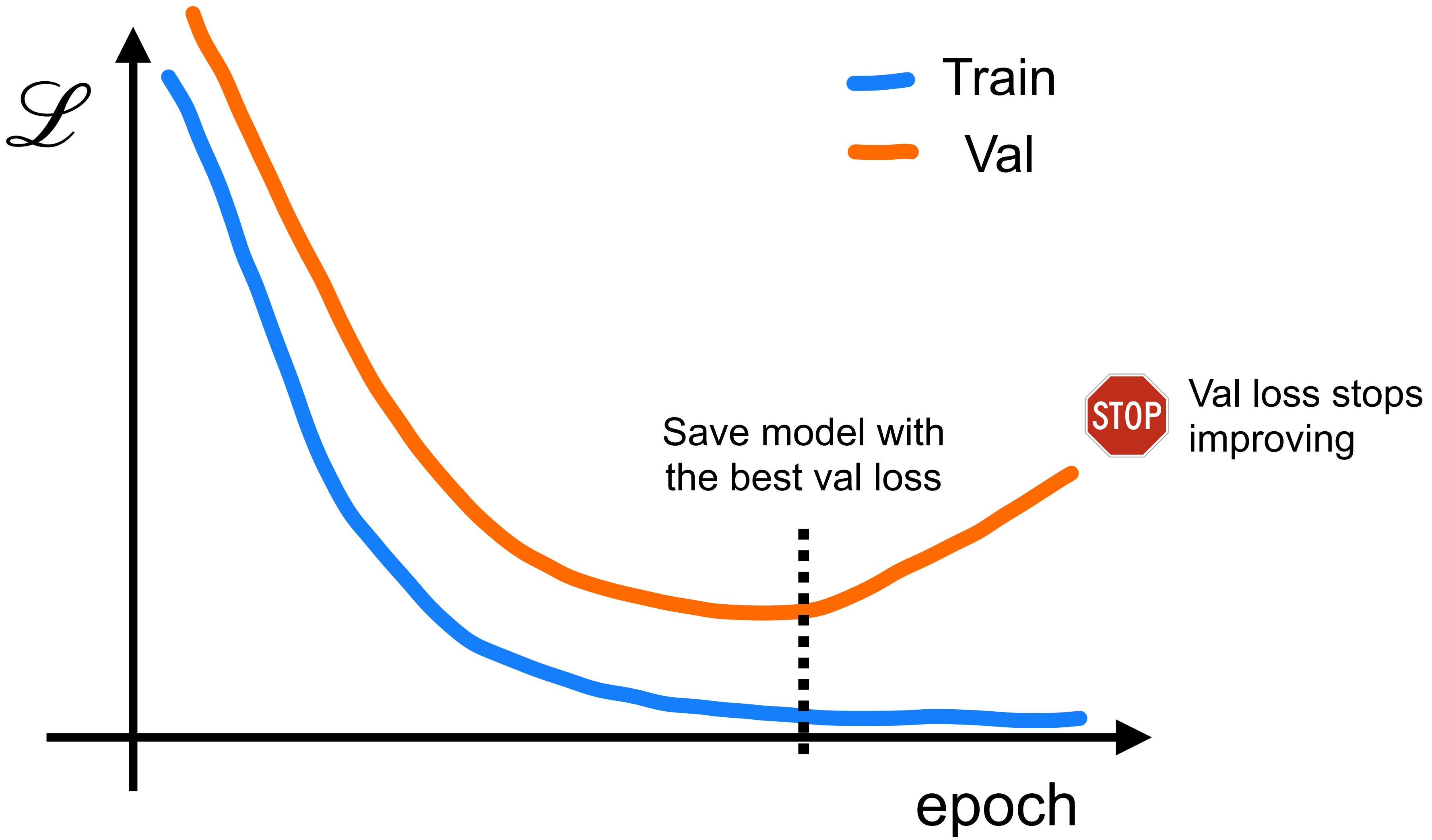
Intimately tied to learning rate!

If you increase the batch size by a factor of 2, scale α by $\frac{1}{2}$ for a fair comparison

Early stopping



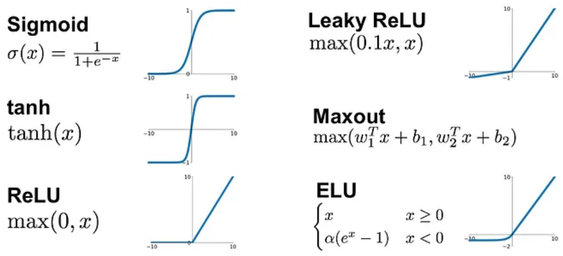
Early stopping



Hyperparameter search

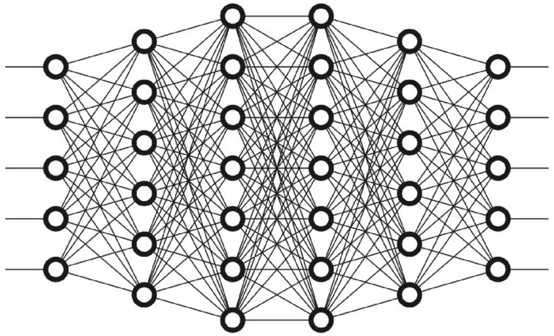
Already many options…

1 Activations



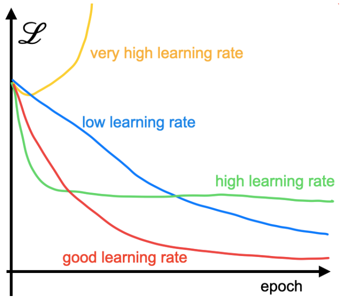
Medium [article]

2 Number of layers



3 Nodes / layer

4 Learning rate



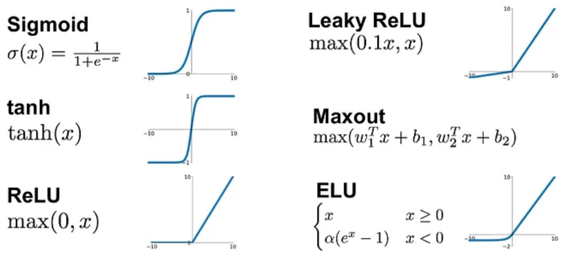
+ many more!

in this talk and others!

Hyperparameter search

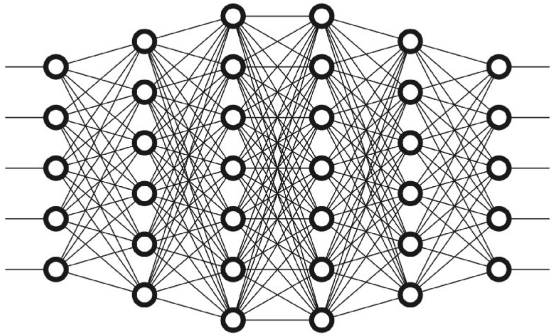
Already many options…

1 Activations



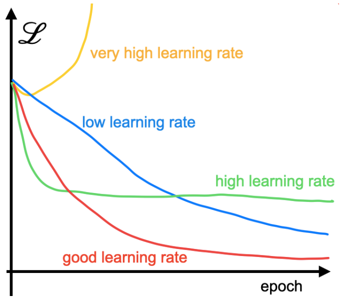
Medium article

2 Number of layers



3 Nodes / layer

4 Learning rate



Coarse scan:

3 activations {sigmoid, ReLU, ELU}
x **3** layers {5, 10, 20}
x **4** nodes {10, 50, 250, 200}
x **4** learning rates {1e-2, 3e-3, 1e-3, 3e-4}
x **2** {with and w/o scheduler}

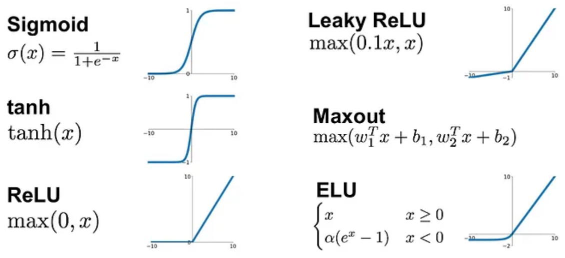
= 384 trainings !!!

+ many more!
in this talk and others!

Hyperparameter search

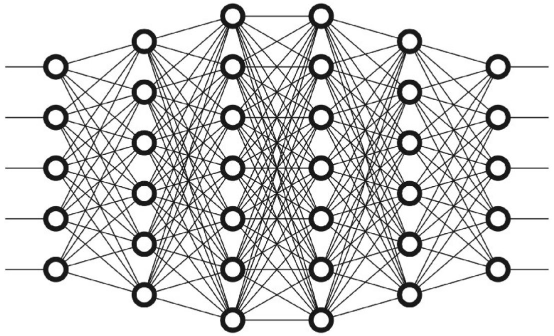
Already many options…

1 Activations



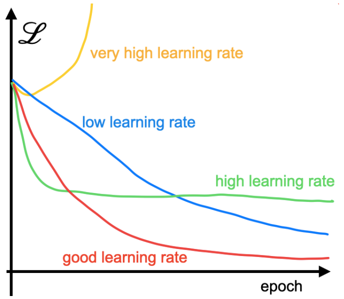
Medium [article]

2 Number of layers



3 Nodes / layer

4 Learning rate



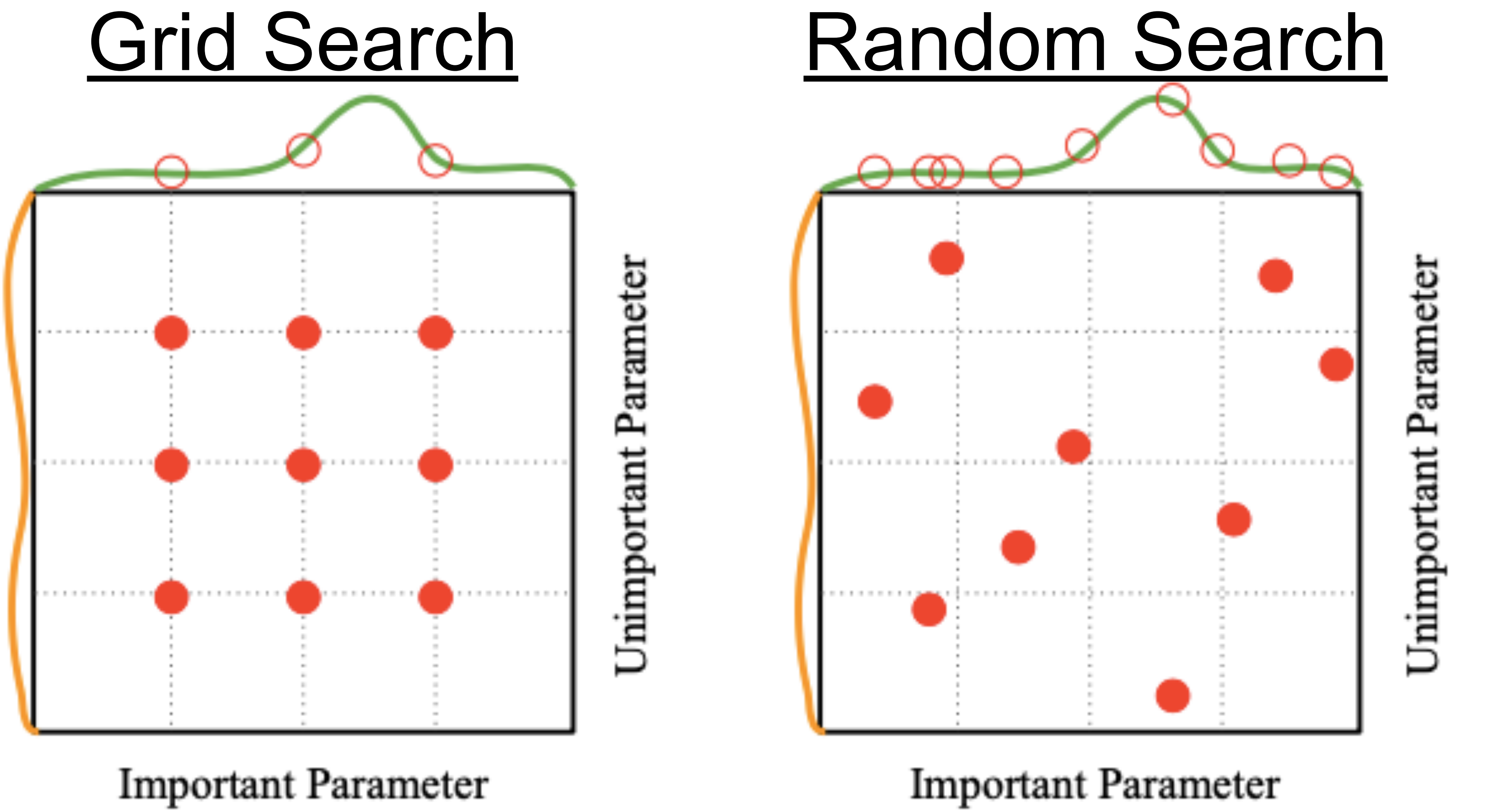
+ many more!

in this talk and others!

Coarse scan:

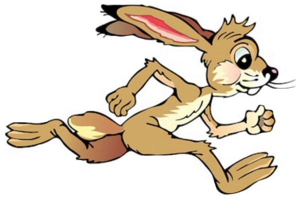
3 activations {sigmoid, ReLU, ELU}
x **3** layers {5, 10, 20}
x **4** nodes {10, 50, 250, 200}
x **4** learning rates {1e-2, 3e-3, 1e-3, 3e-4}
x **2** {with and w/o scheduler}
x **10** K-fold cross validation (K=10)
= 3840 trainings !!!

Hyper-parameter search



Not all hyper-parameters are equal!

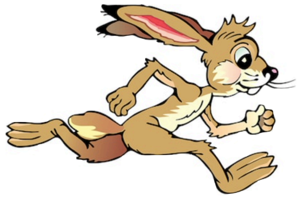
Hyperparameter strategies



Fast prototyping

1. Start with a subset of the training dataset
2. Find parameters for a model that overfits
3. Start the random search around this point

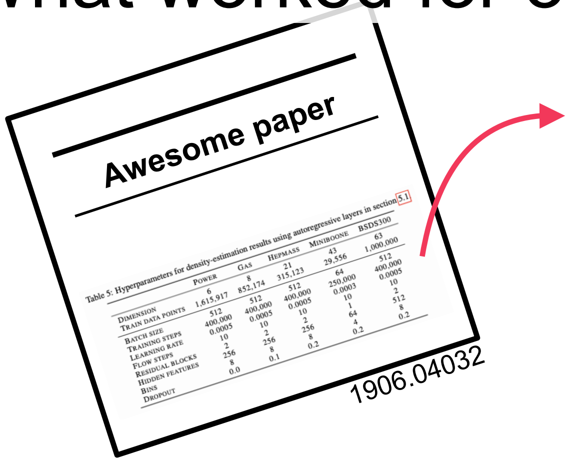
Hyperparameter strategies



Fast prototyping

1. Start with a subset of the training dataset
2. Find parameters for a model that overfits
3. Start the random search around this point

What worked for others?



Awesome paper

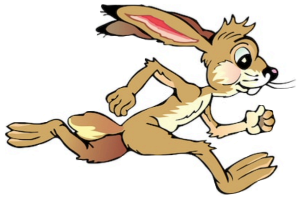
Table 5: Hyperparameters for density-estimation results using autoregressive layers in section 5.1

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
DIMENSION	6	8	21	43	63
TRAIN DATA POINTS	1,615,917	852,174	315,123	29,556	1,000,000
BATCH SIZE	512	512	512	64	512
TRAINING STEPS	400,000	400,000	400,000	250,000	400,000
LEARNING RATE	0.0005	0.0005	0.0005	0.0003	0.0005
FLOW STEPS	10	10	10	10	10
RESIDUAL BLOCKS	2	2	2	1	2
HIDDEN FEATURES	256	256	256	64	512
BINS	8	8	8	4	8
DROPOUT	0.0	0.1	0.2	0.2	0.2

1906.04032

Starting point random search

Hyperparameter strategies



Fast prototyping

1. Start with a subset of the training dataset
2. Find parameters for a model that overfits
3. Start the random search around this point

What worked for others?

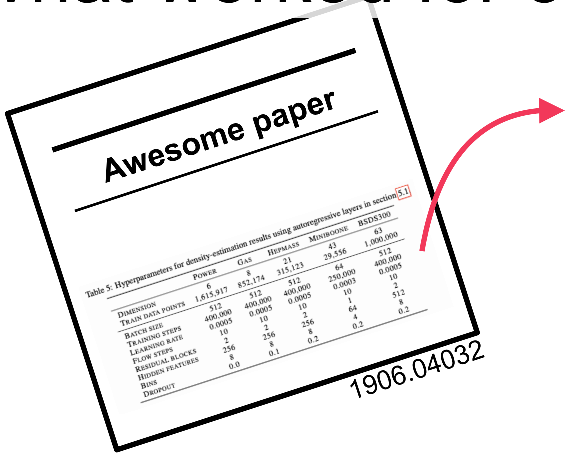


Table 5: Hyperparameters for density-estimation results using autoregressive layers in section 5.1

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
DIMENSION	6	8	21	43	63
TRAIN DATA POINTS	1,615,917	852,174	315,123	29,556	1,000,000
BATCH SIZE	512	512	512	64	512
TRAINING STEPS	400,000	400,000	400,000	250,000	400,000
LEARNING RATE	0.0005	0.0005	0.0005	0.0003	0.0005
FLOW STEPS	10	10	10	10	10
RESIDUAL BLOCKS	2	2	2	1	2
HIDDEN FEATURES	256	256	256	64	512
BINS	8	8	8	4	8
DROPOUT	0.0	0.1	0.2	0.2	0.2

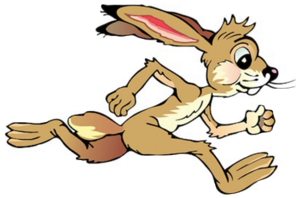
Starting point random search

Scan in log space

Coarse scan:

3 activations {sigmoid, ReLU, ELU}
x **3** layers {5, 10, 20}
x **4** nodes {10, 50, 250, 200}
x **4** learning rates {1e-2, 3e-3, 1e-3, 3e-4}
x **2** {with and w/o scheduler}
x **10** K-fold cross validation (K=10)
= 3840 trainings !!!

Hyperparameter strategies



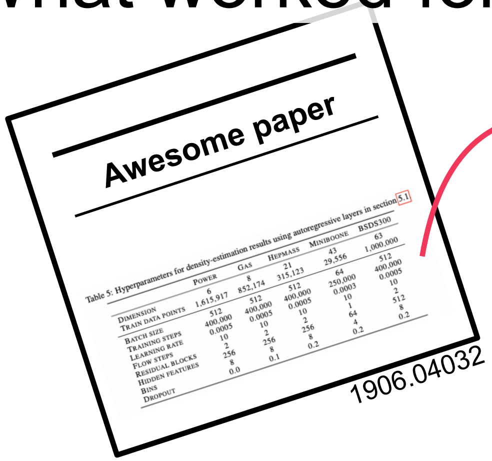
Fast prototyping

1. Start with a subset of the training dataset
2. Find parameters for a model that overfits
3. Start the random search around this point

Can automate!

OPTUNA

What worked for others?



Starting point random search

Scan in log space

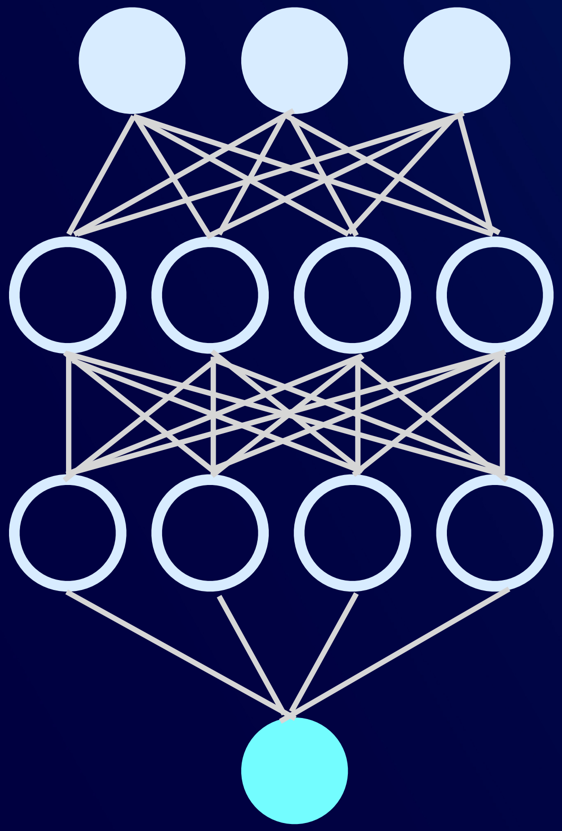
Coarse scan:

3 activations {sigmoid, ReLU, ELU}
x **3** layers {5, 10, 20}
x **4** nodes {10, 50, 250, 200}
x **4** learning rates {1e-2, 3e-3, 1e-3, 3e-4}
x **2** {with and w/o scheduler}
x **10** K-fold cross validation (K=10)
= 3840 trainings !!!

Starting off...

✋ Would love feedback + discussions!

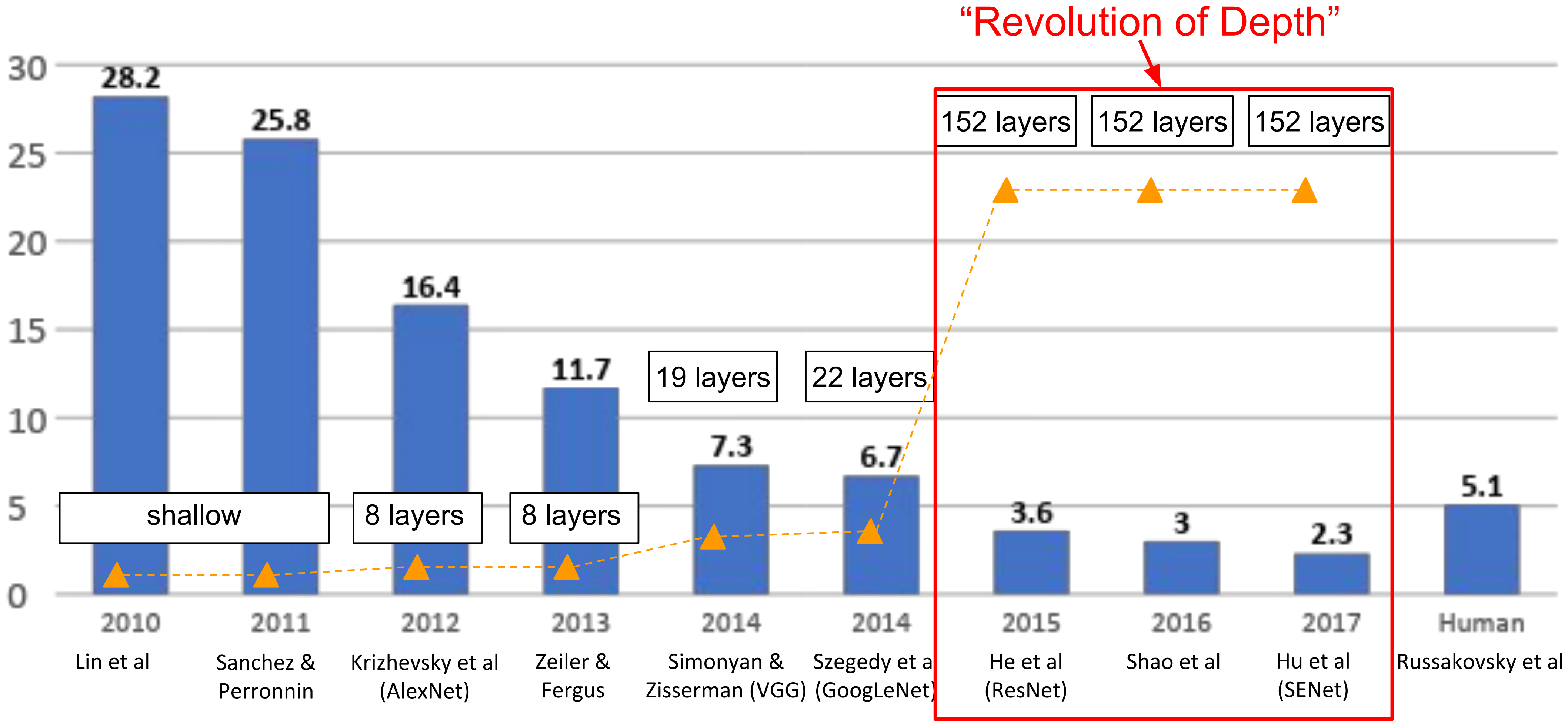
Models in the Deep Learning Era



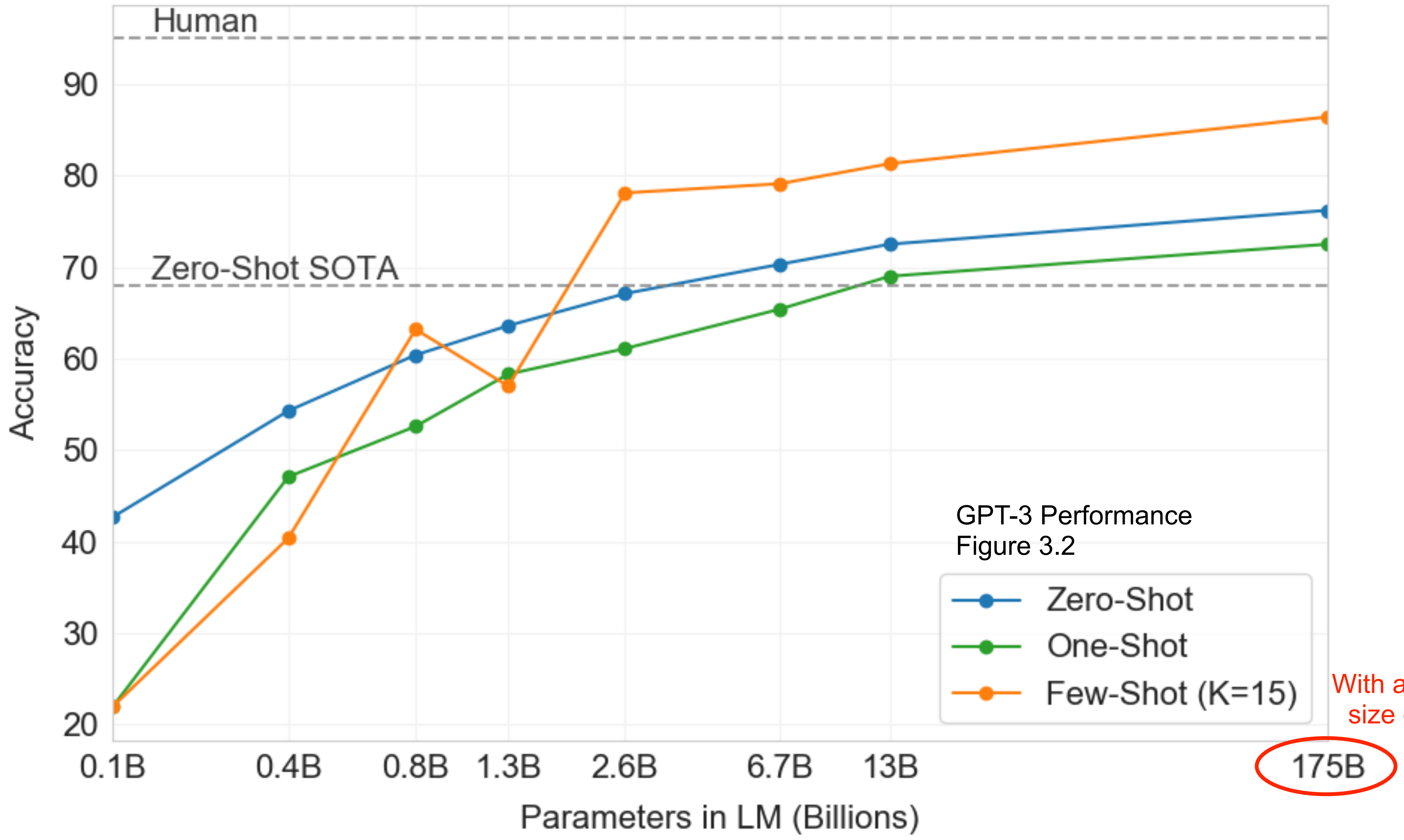
Slide from CS231n Lecture 9

The Deep Learning Revolution

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Natural Language Processing



GPT-3 Performance
Figure 3.2

With a training dataset size of 500B words.

Deep Learning philosophy...

Bigger = better

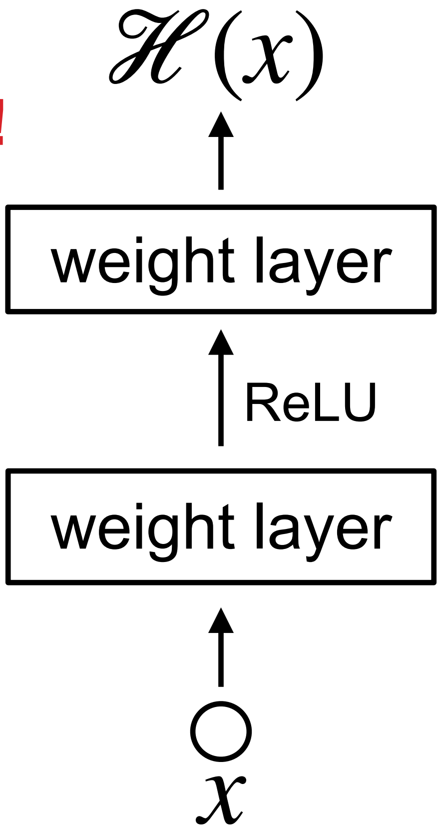
Deep Learning philosophy…

Bigger = better

ResNet building block

If layer isn't needed,
need to learn the identity.

Degrades performance!



Normal layers

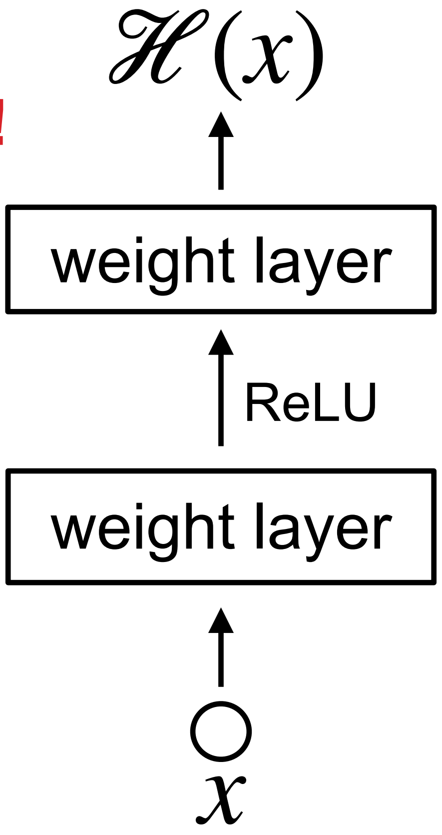
ResNet building block

If layer isn't needed,
need to learn the identity.

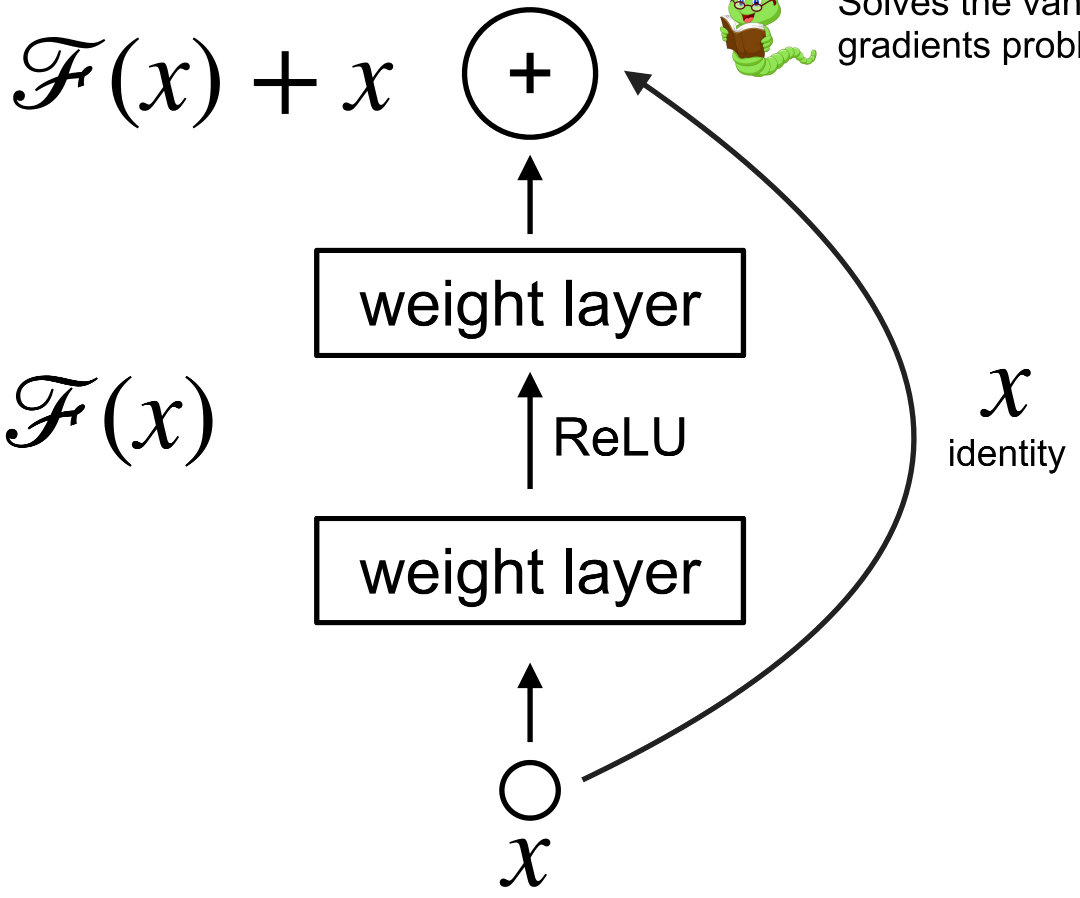
Learn the correction factor

Solves the vanishing gradients problem!

Degrades performance!



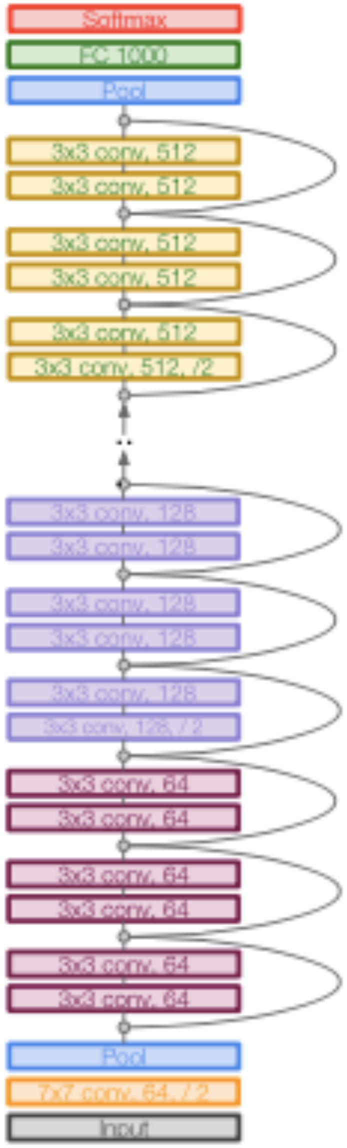
Normal layers



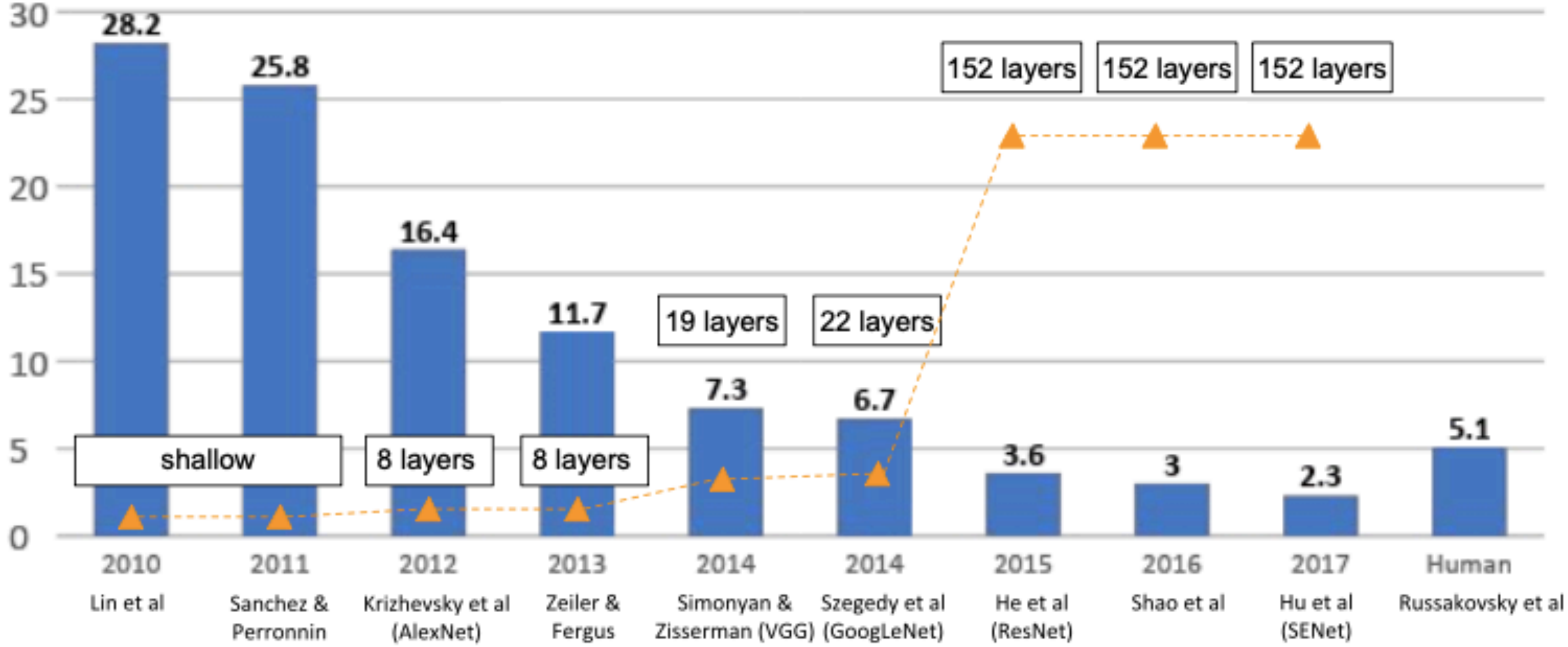
Residual block

ResNet: performance

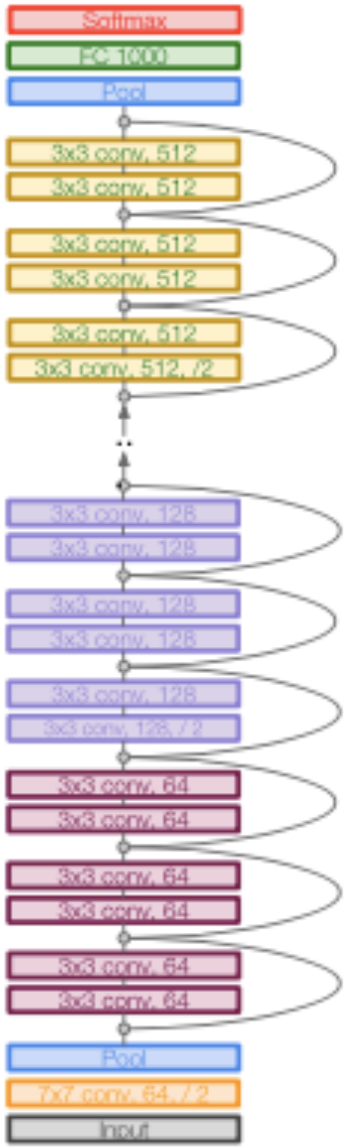
Learn the correction factor



What do we gain???

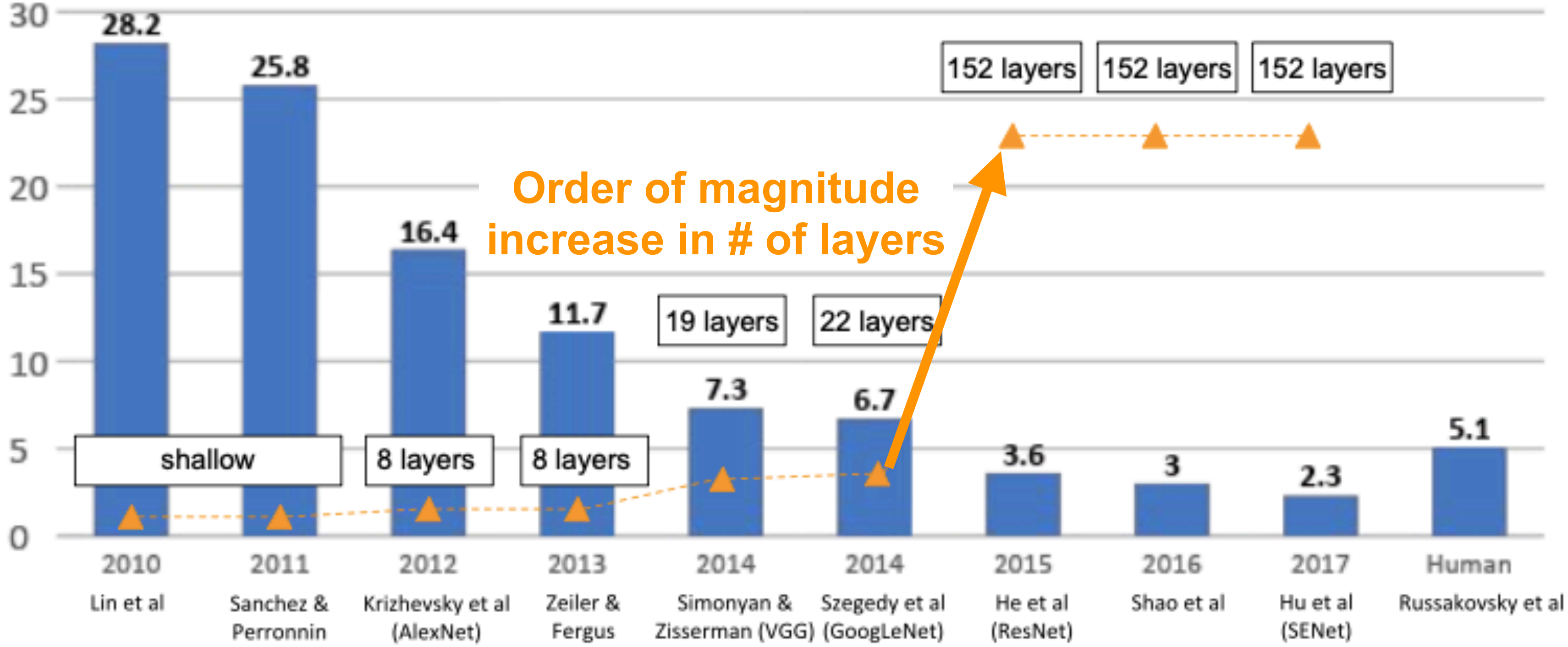


ResNet: performance

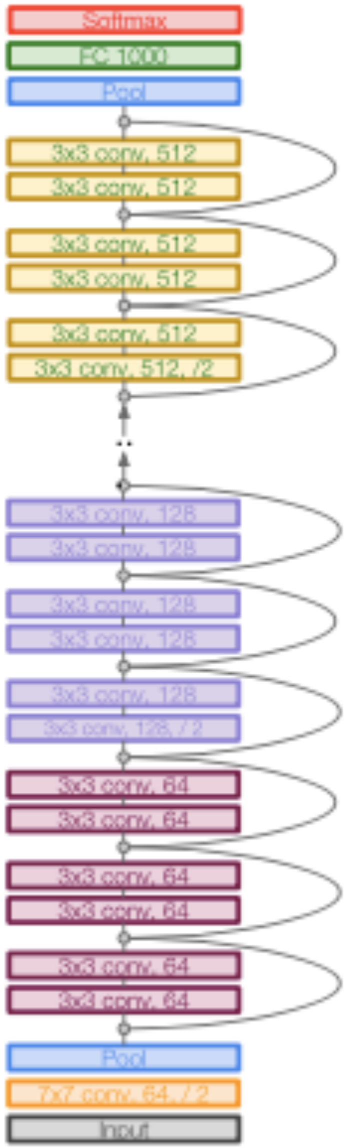


Learn the correction factor

What do we gain???

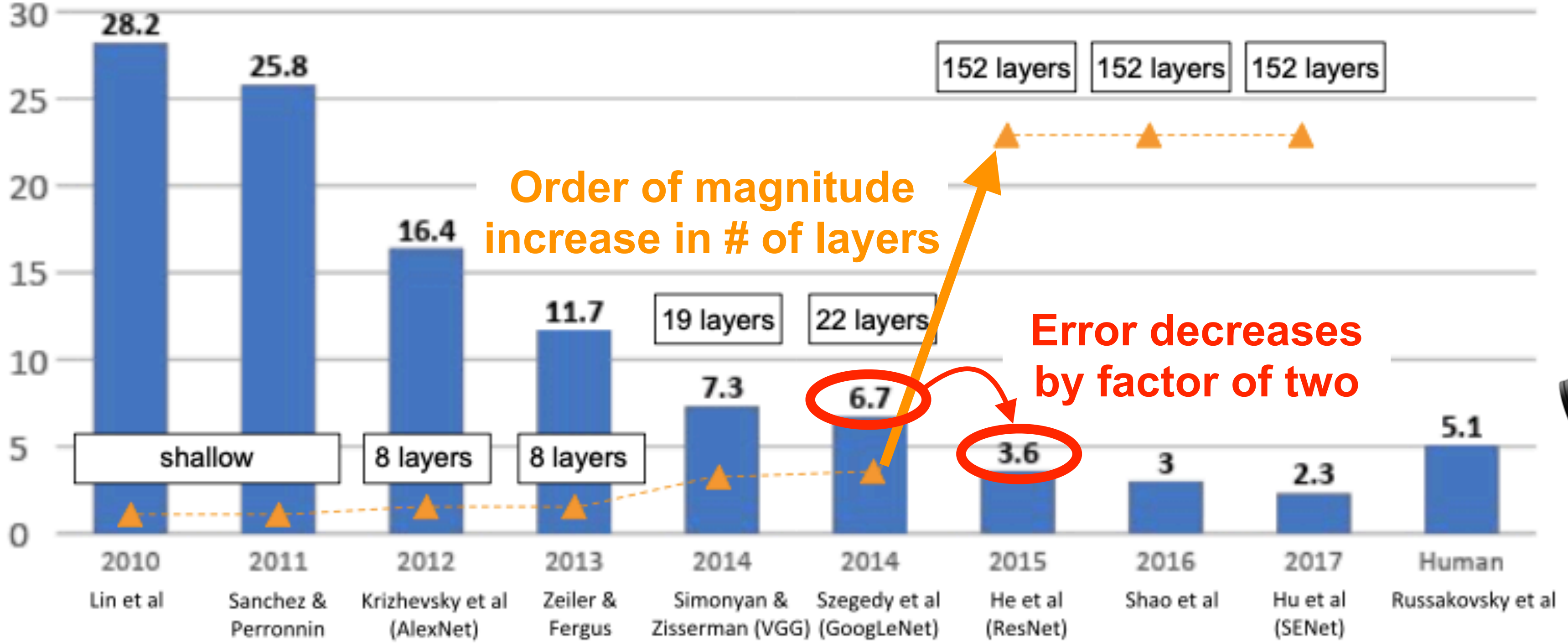


ResNet: performance

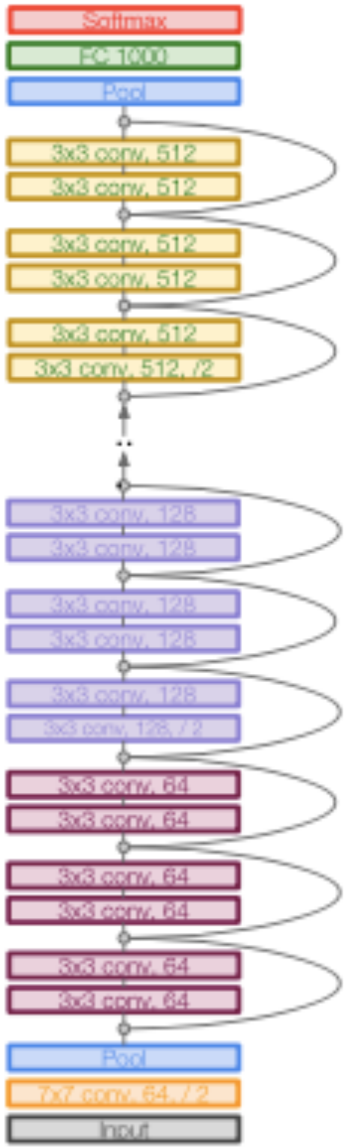


Learn the correction factor

What do we gain???

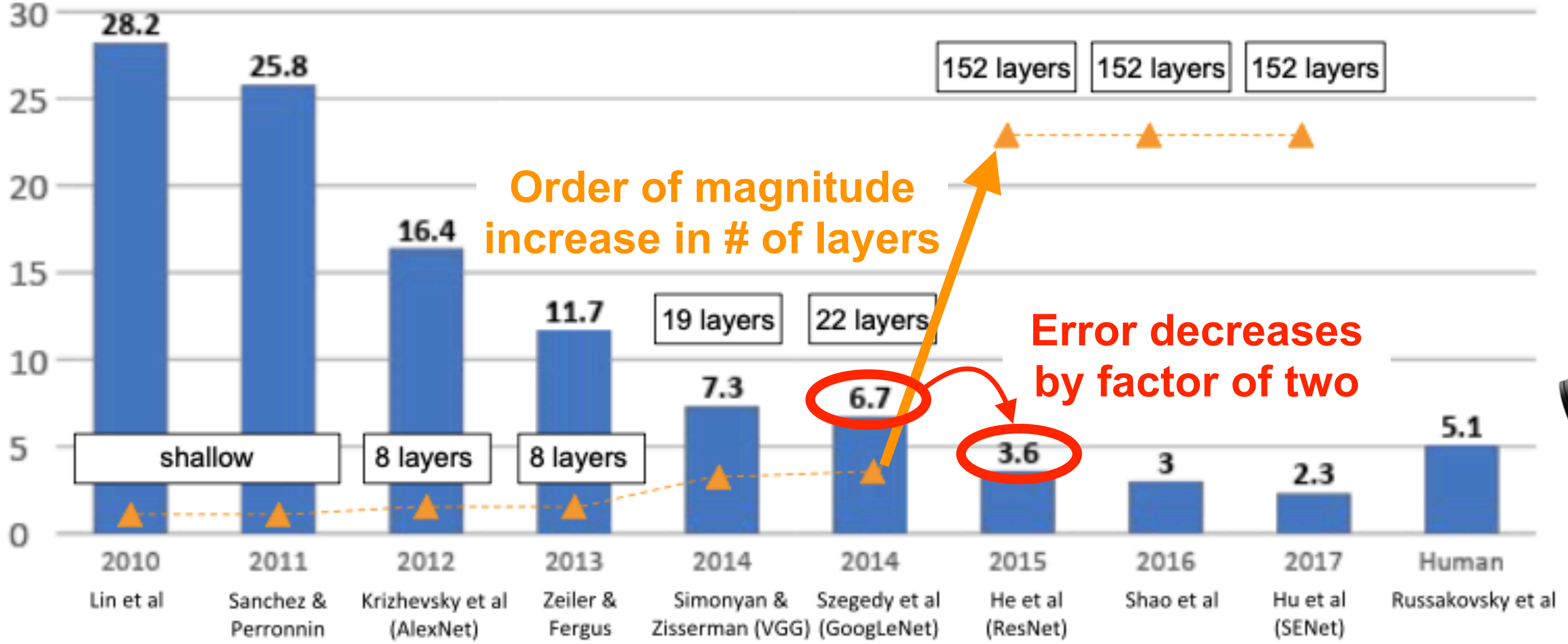


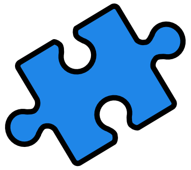
ResNet: performance



Learn the correction factor

What do we gain???





Crucial for almost every modern architecture from natural language (1706.03762) to generative models (1906.04032).

Regularization Techniques

How to restrict the optimization problem to help the NN generalize better.

Regularization

$$\mathcal{L}_\beta(w) = \mathcal{L}(w) + \beta\, \mathcal{R}(w)$$

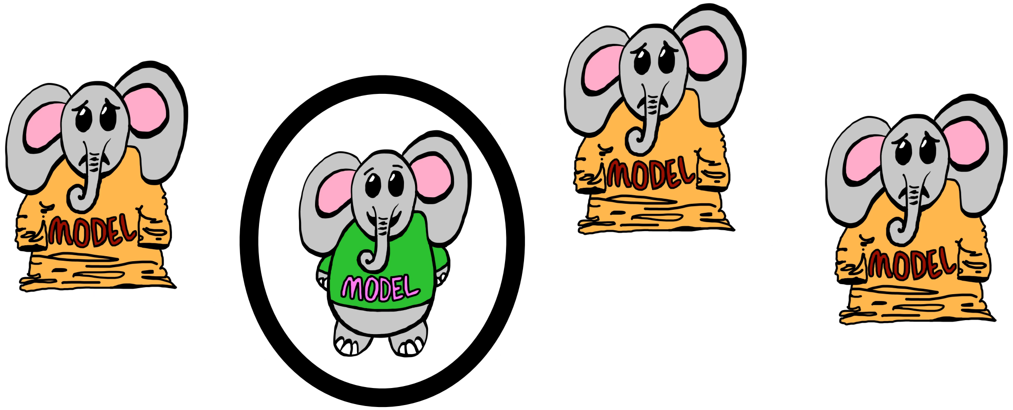
Fit the training data

Penalize complicated models

Hyperparameter governing tradeoff of the two objectives.

Occam's razor for ML

When multiple models describe the training data… choose the simplest one!



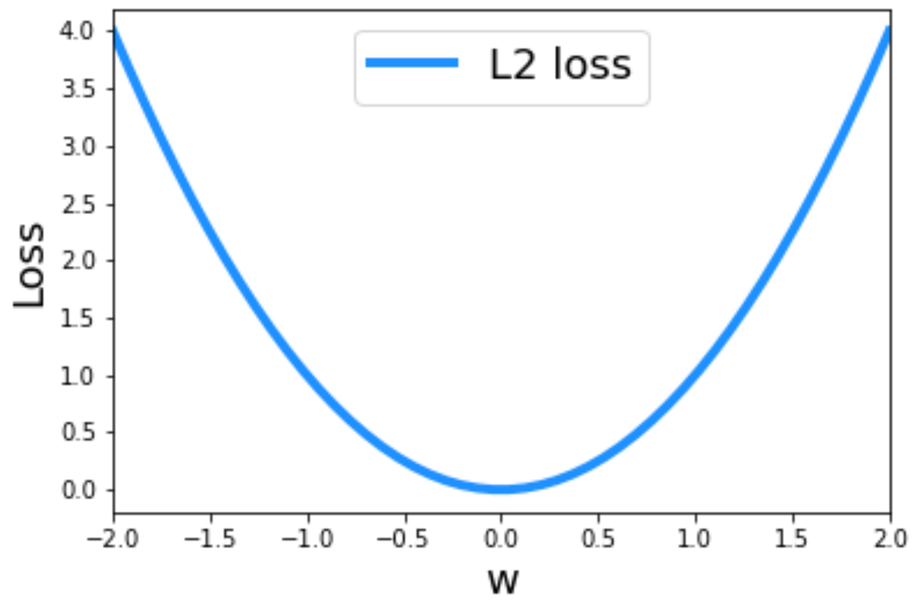
L2 Regularization (most common for NNs)

$$\mathcal{L}_\beta(w) = \mathcal{L}(w) + \beta\ |w|^2$$

Encourages weights to be *small*

"Weight decay"

$$\begin{aligned} w &= w - \alpha \nabla_w \mathcal{L}_\lambda \\ &= w - \alpha \nabla_w \left(\mathcal{L} + \beta w^2\right) \\ &= \underbrace{(1 - 2\beta)}_{\text{Decay}} w - \alpha \nabla_w \mathcal{L} \end{aligned}$$

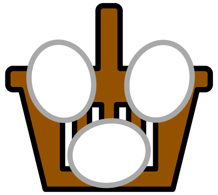


✅ Include in random search

(Log scale, e.g, β = 0, 1e-6, 1e-4)

Dropout: intro

Issue: Don't want the NN to rely heavily on individual features

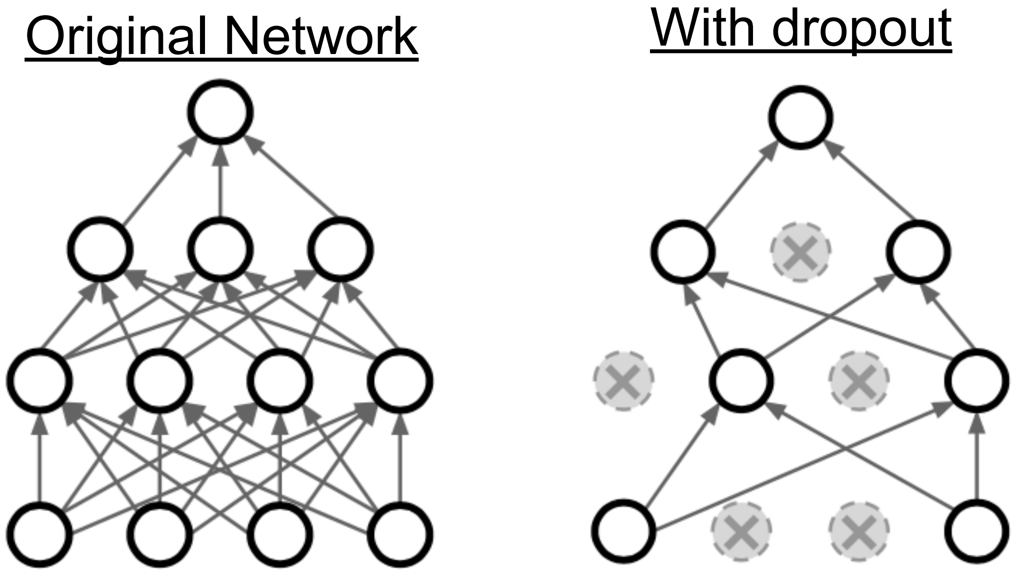


ML version of not putting all your eggs in one basket

Srivastava et al, "Dropout: A simple way to prevent neural networks from overfitting", JMLR 2014
CS231n Lecture 7

Dropout: idea

Issue: Don't want the NN to rely heavily on individual features

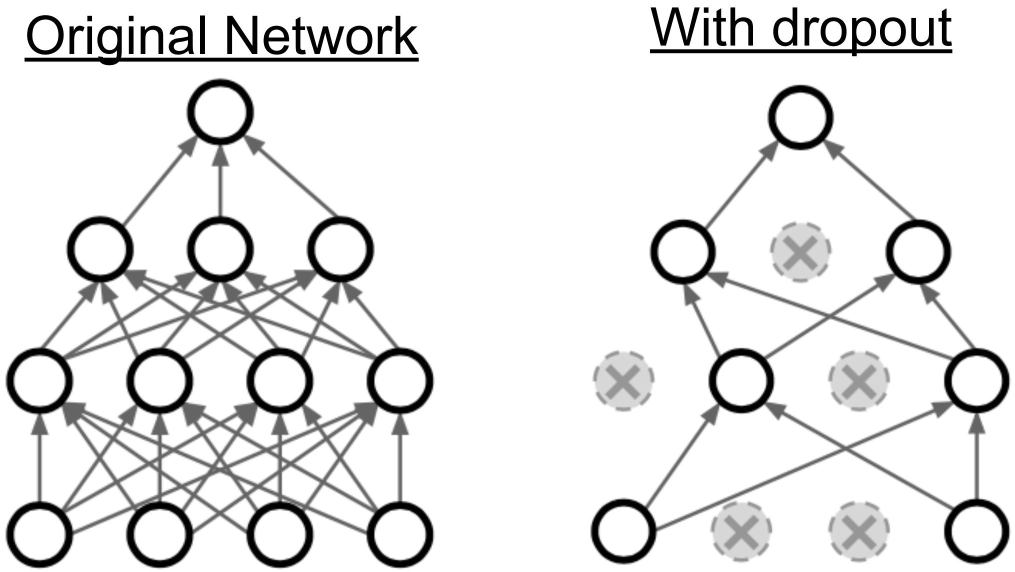


At **training time**, zero out some neurons with dropout fraction p

Hyperparameter we need to optimize!

Dropout: idea

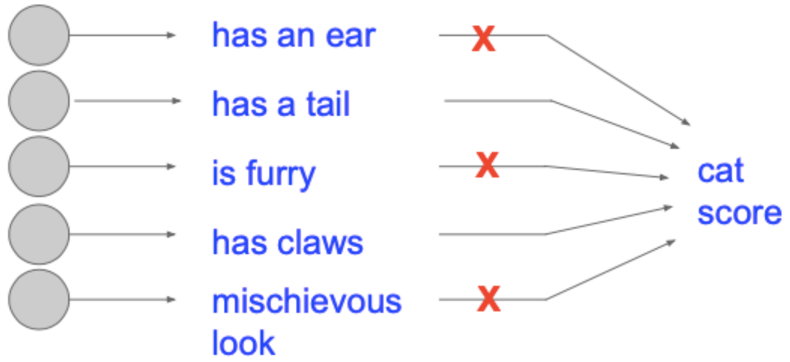
Issue: Don’t want the NN to rely heavily on individual features



At **training time**, zero out some neurons with dropout fraction p

Hyperparameter we need to optimize!

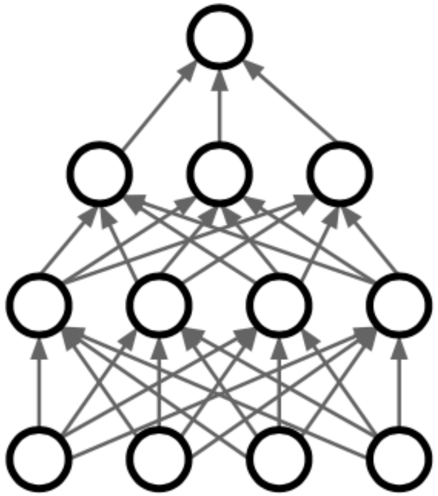
✅ Encourages learning robust features



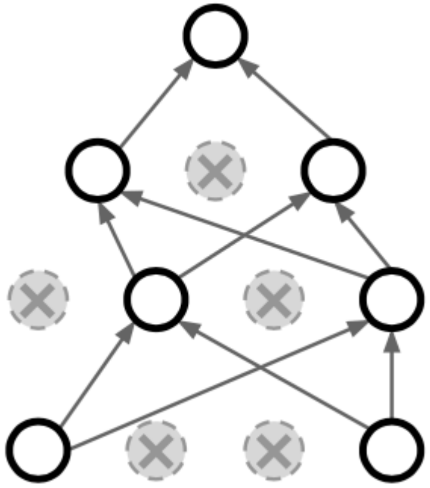
Dropout: idea

Issue: Don't want the NN to rely heavily on individual features

Original Network



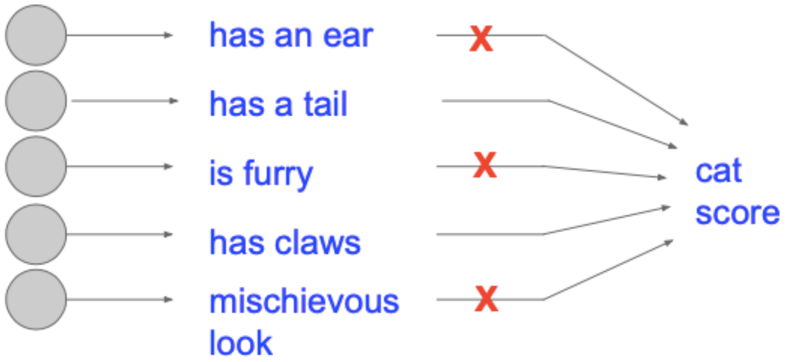
With dropout



At **training time**, zero out some neurons with dropout fraction p

Hyperparameter we need to optimize!

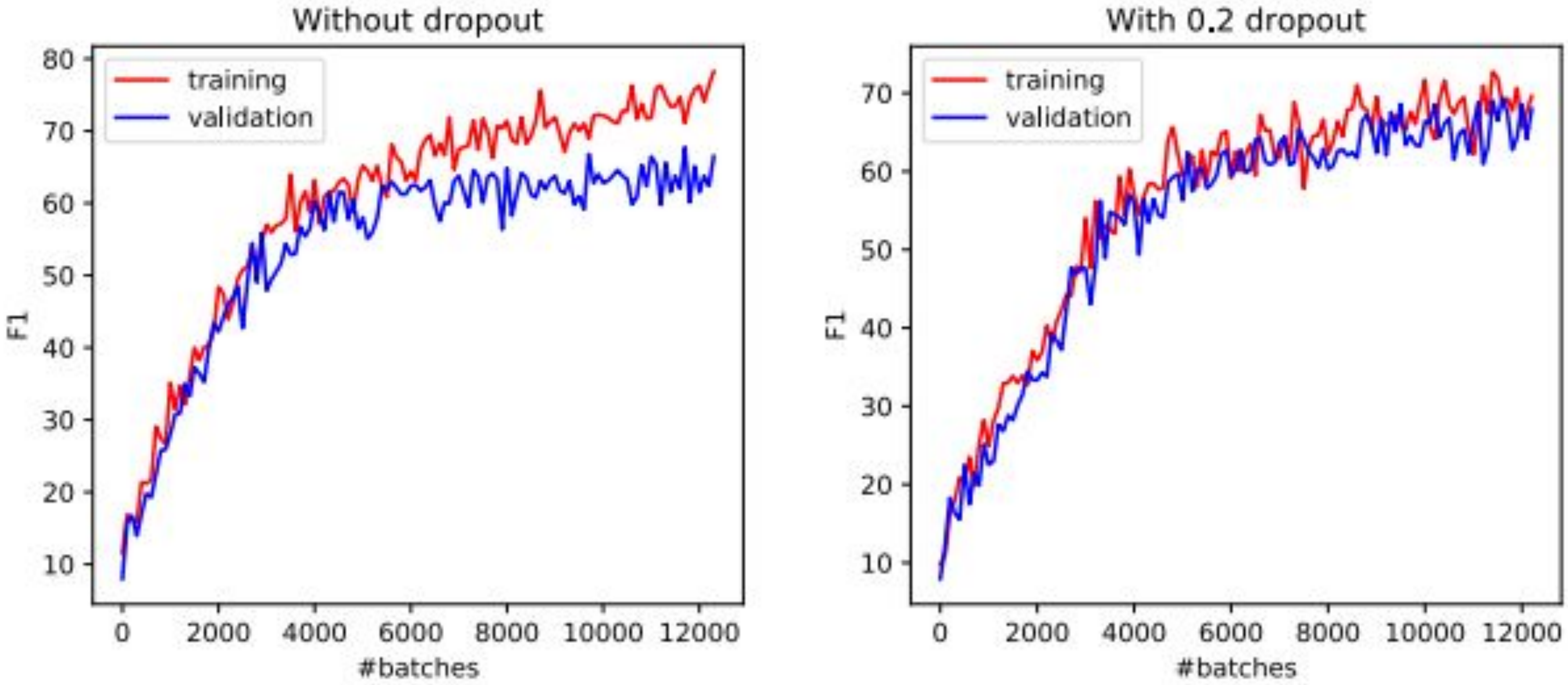
✅ Encourages learning robust features



At **test time**, use all the neurons for prediction!!

Like training an ensemble of the NNs without being as €€€

Dropout: performance

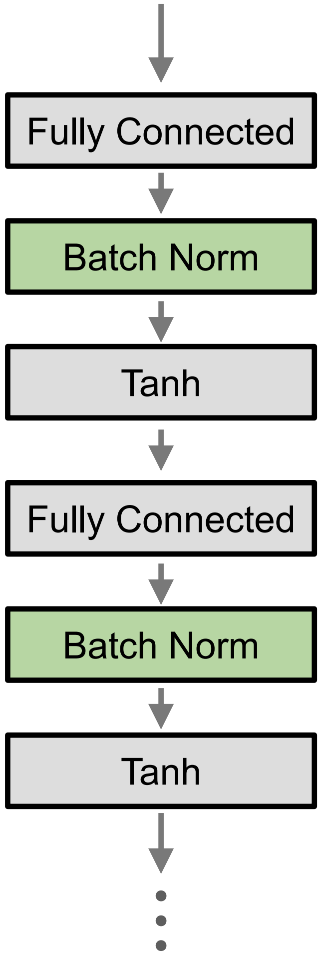


Overfit first — then try to close the gap!!

Batch Normalization

What about the features in the hidden layers?

"You want zero mean and unit variance operations? Just make them so!"



At **training time**, normalize over the activations of the minibatch

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}\left[x^{(k)}\right]}{\sqrt{\text{Var}\left[x^{(k)}\right]}}$$

Additional learnable parameters for the scale and shift: γ and β.

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

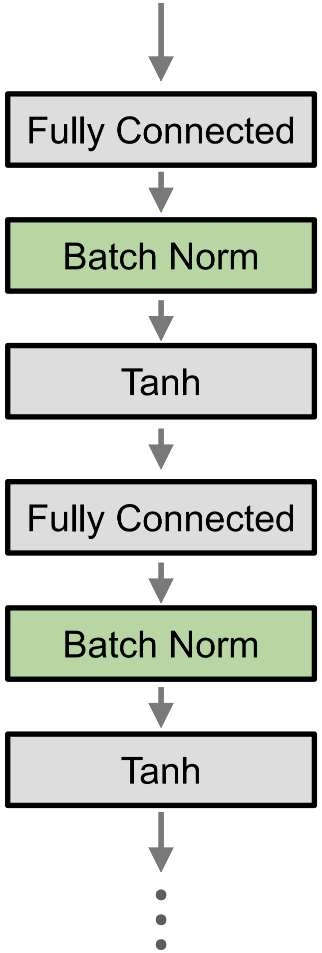
Original paper: inserted after Fully Connected layers, and before nonlinearity.

1502.03167
Stanford's CS 231n [Lecture 6]

Batch Normalization

What about the features in the hidden layers?

“You want zero mean and unit variance operations? Just make them so!”



At **training time**, normalize over the activations of the minibatch

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}\left[x^{(k)}\right]}{\sqrt{\text{Var}\left[x^{(k)}\right]}}$$

Additional learnable parameters for the scale and shift: γ and β.

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Original paper: inserted after Fully Connected layers, and before nonlinearity.

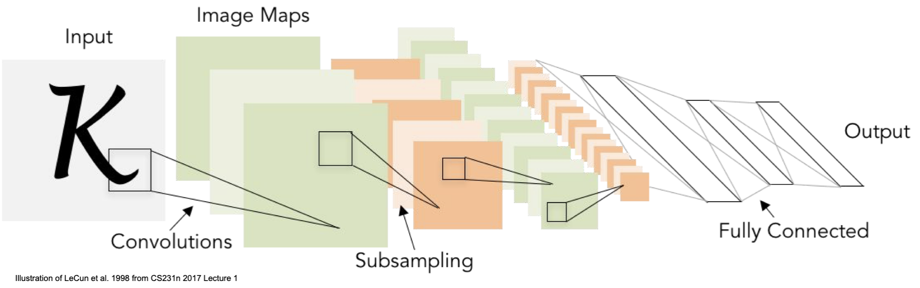
For set / sequence data, [LayerNorm] also useful

Weight sharing

Alternative architectures designed to *reuse weights* as suited for the input data.

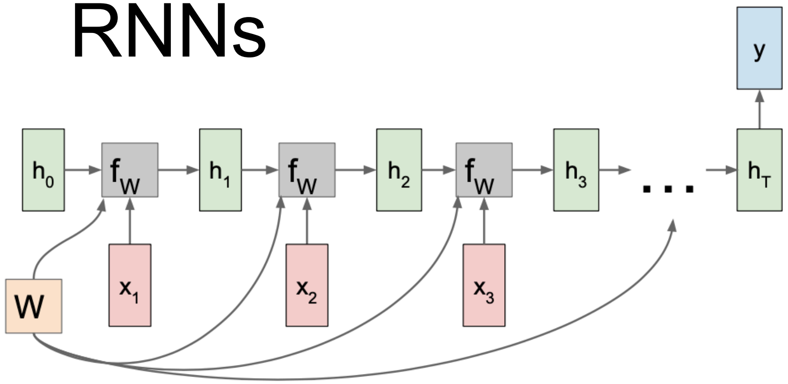
“Don’t relearn what you don’t need to”

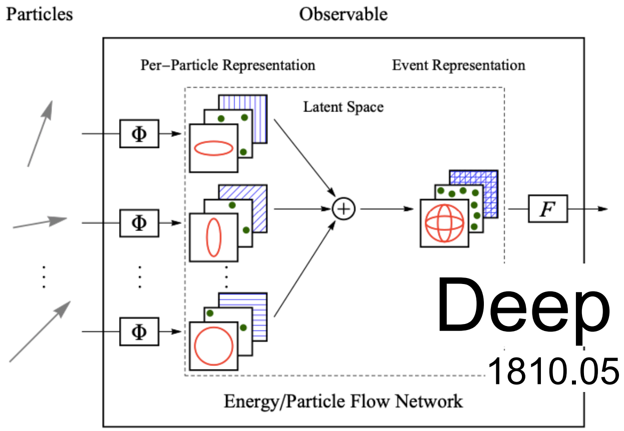
CNNs



Talk tomorrow by Till!

RNNs



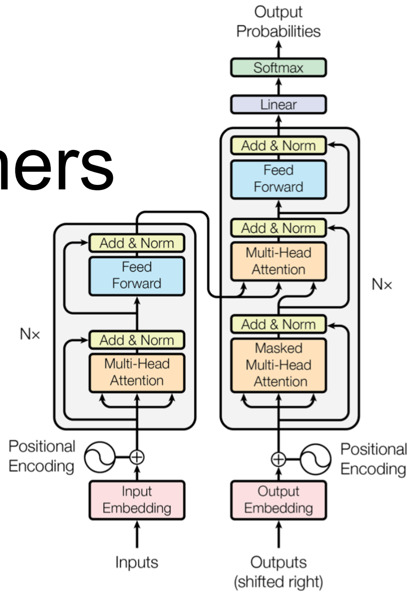


Deep Sets

1810.05165

Transformers

1706.03762

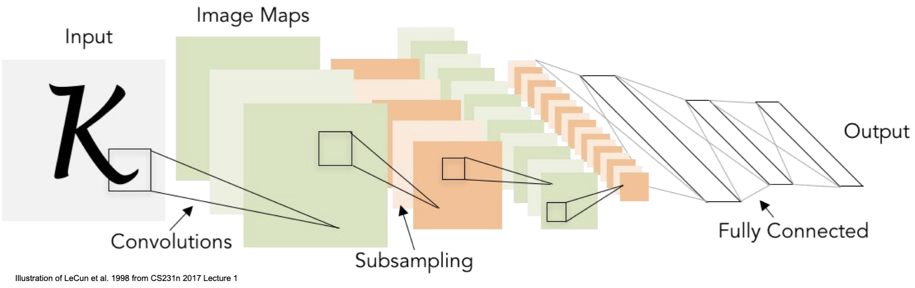


Weight sharing

Alternative architectures designed to *reuse weights* as suited for the input data.

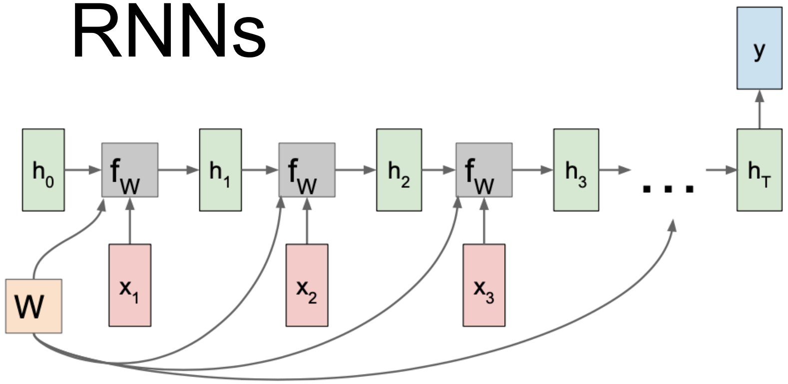
"Don't relearn what you don't need to"

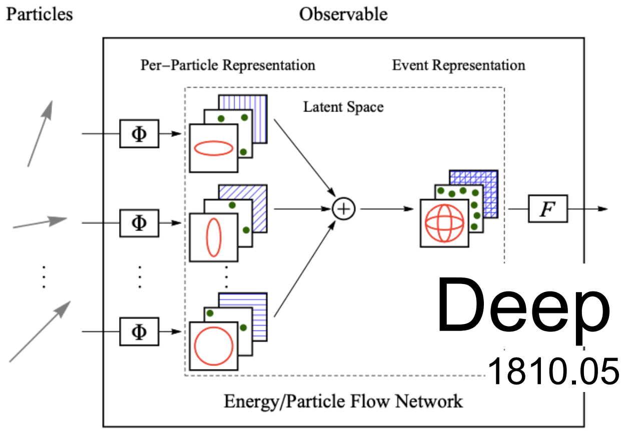
CNNs



Talk tomorrow by Till!

RNNs



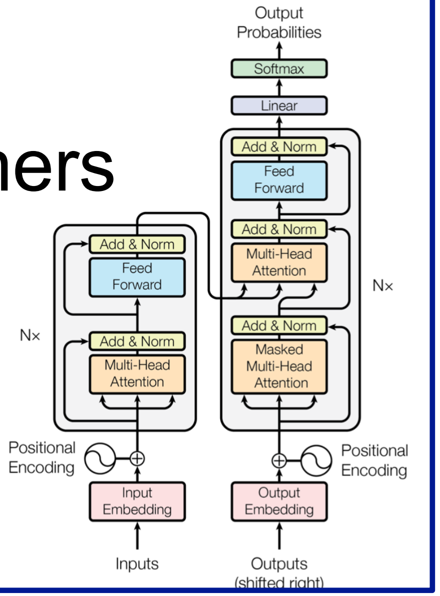


Deep Sets

1810.05165

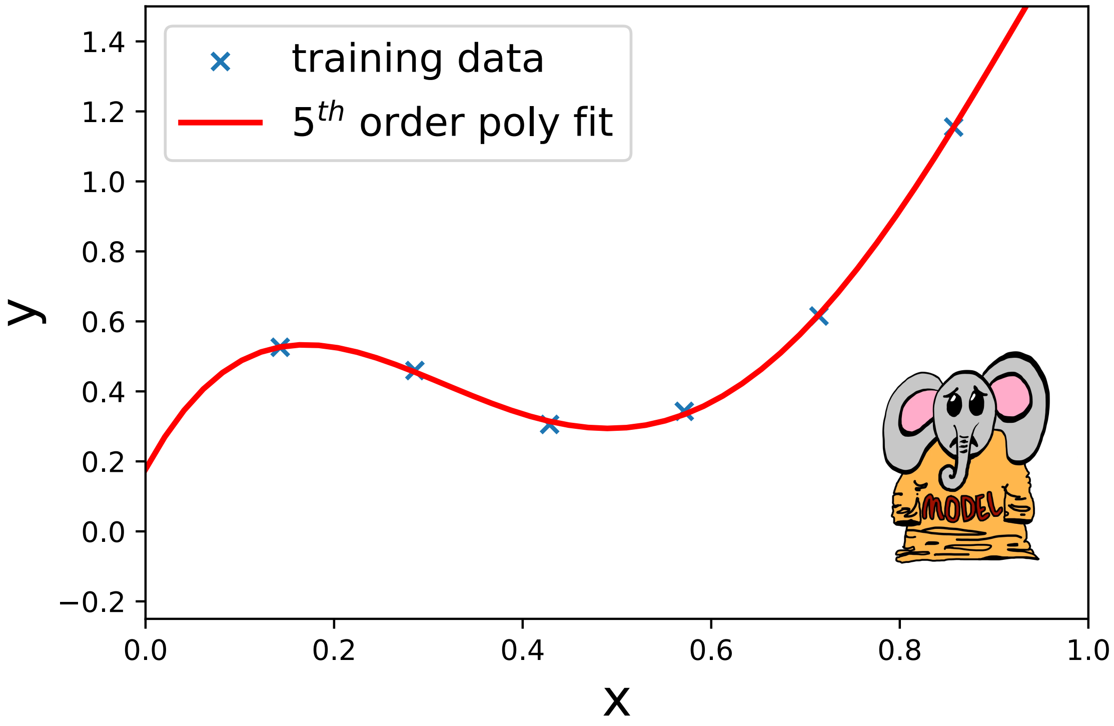
Transformers

1706.03762

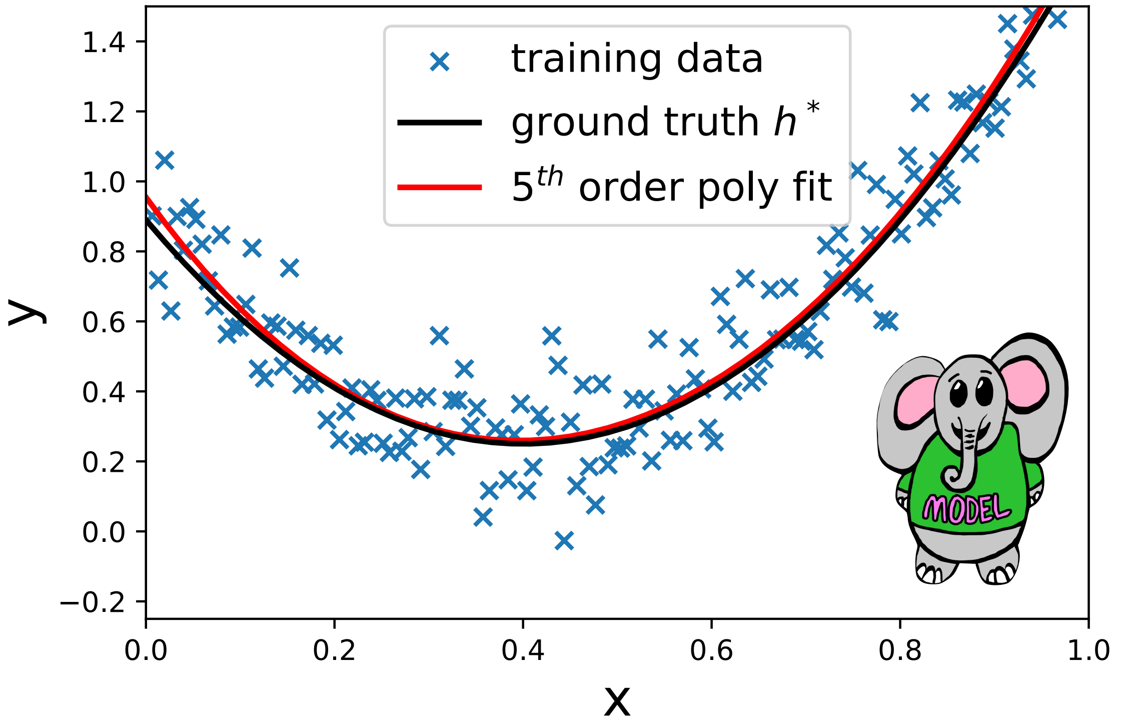


Data augmentation : motivation

Recall: More training data <u>reduces variance</u>.



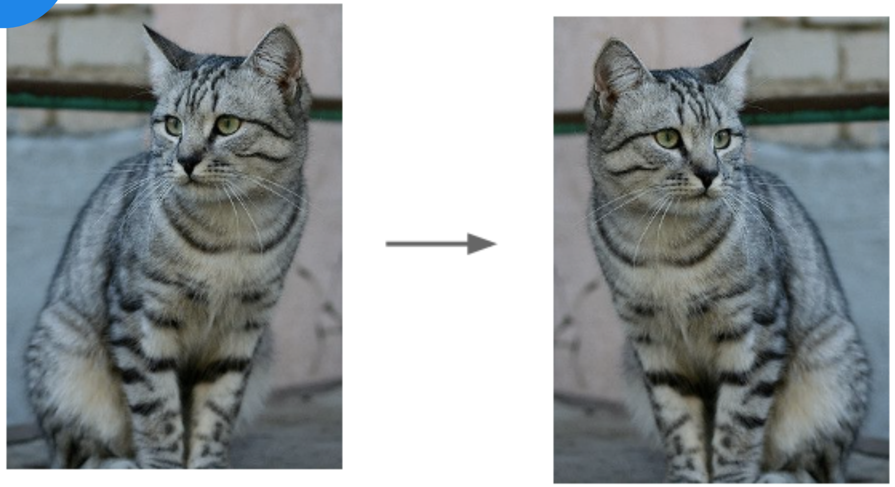
Fit 5^{th} order polynomial on large dataset



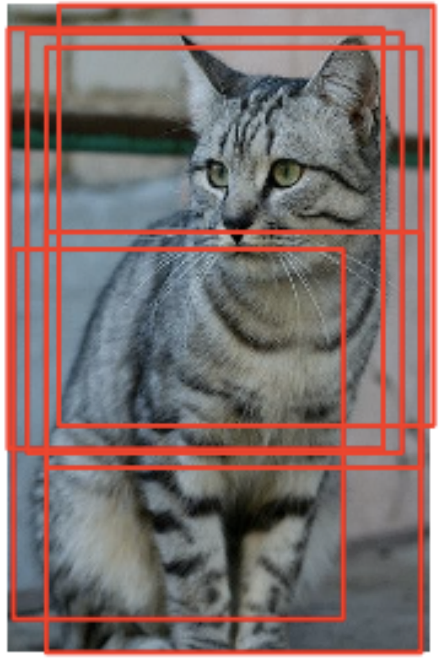
Q: How can you modify your training data to artificially increase your dataset size?

Data augmentation : examples

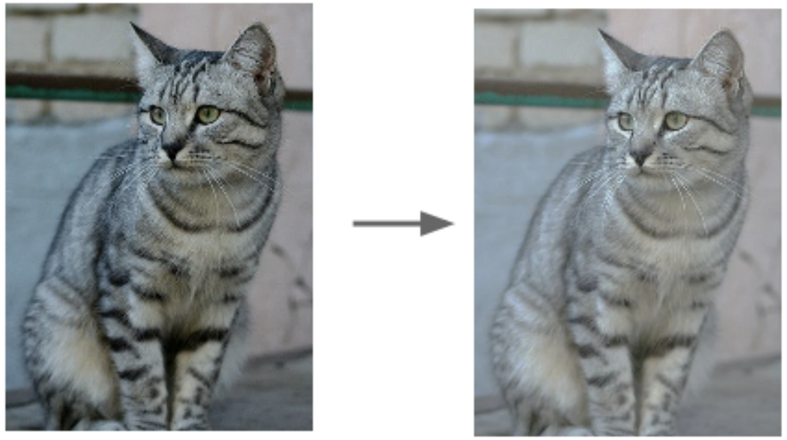
1 Horizontal flips



2 Random crops and scales

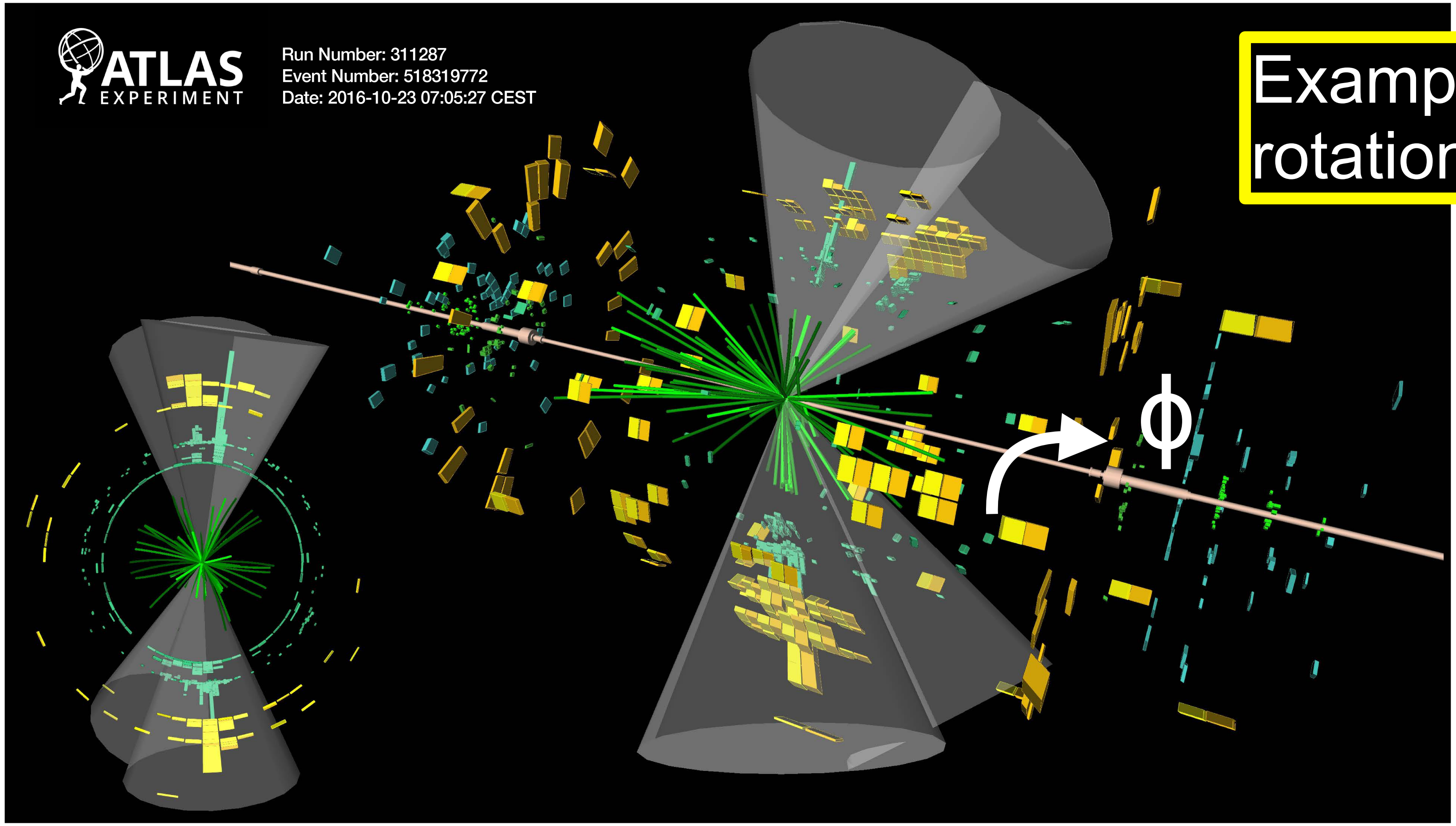


3 Adjust contrast and brightness



Fun fact: the internet isn't big enough for training LLMs anymore! Current research also is adding data augmentations to the training.
— Tip from Oleg Filatov

Data augmentation — physics



Example: Azimuthal rotations

Less used in practice…

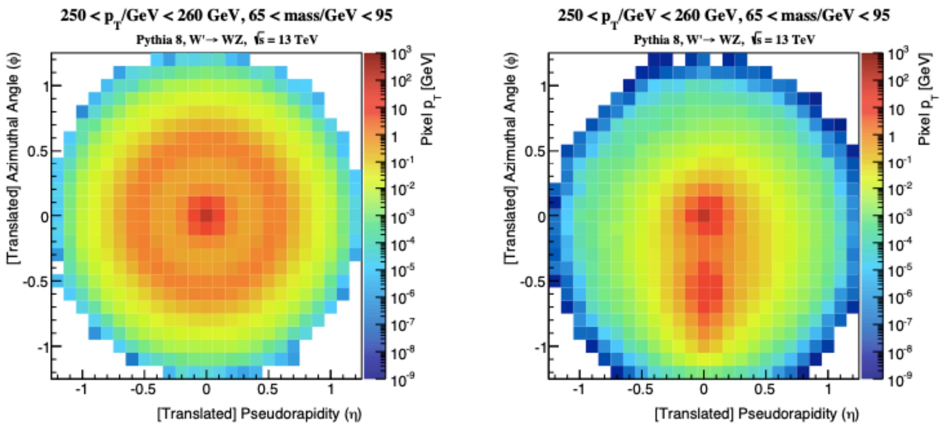
With a simulator and we can often get as many training examples as we want

Data augmentation — alternatives

Issue: Larger models with more data take longer to train

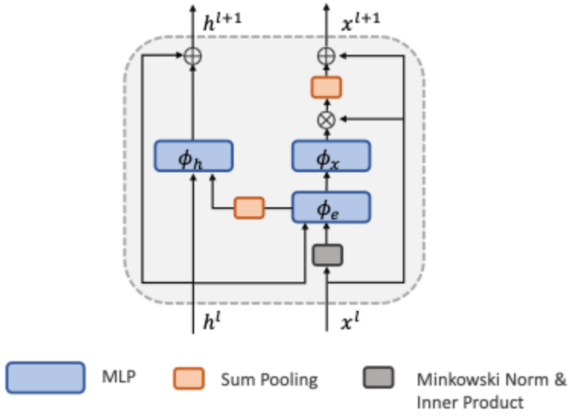
➡ Remove the variation to train faster.

1 Preprocess for uniformity



Jet Images — Deep Learning Edition 1511.05190

2 Architecture design to preserve invariants

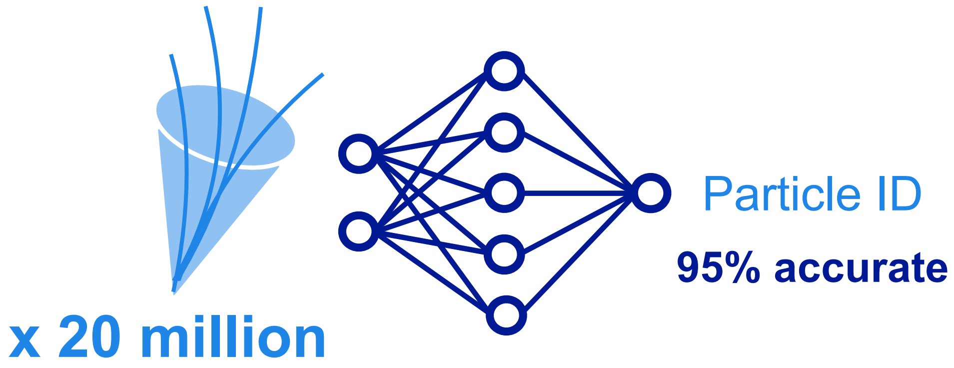


Lorentz Group Equivariant Block (LGEB)

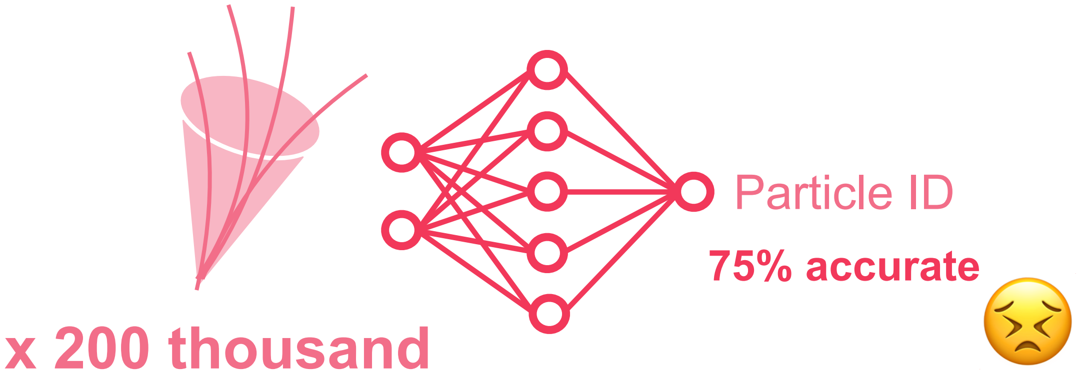
L-GATr: 2405.14806 and Jonas's talk
LorentzInvariance: Lorentz Net and ParT 2202.03772
Azimuthal Symmetry: 2107.02908
+ many others

Fine-tuning / transfer learning

Original Task



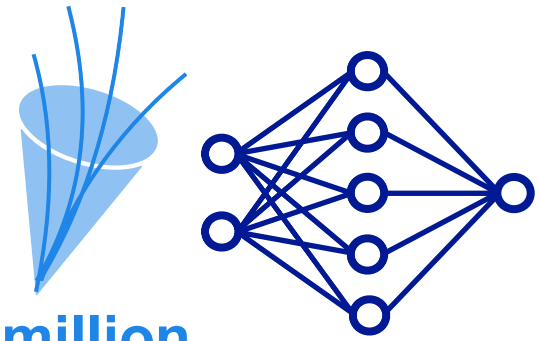
Modified task



Actual example from my PhD!!

Fine-tuning / transfer learning

Original Task

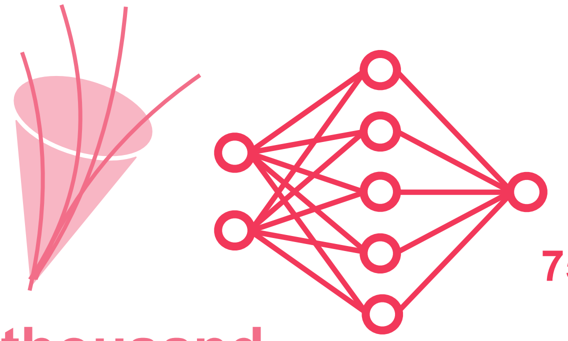


Particle ID

95% accurate

x 20 million

Modified task

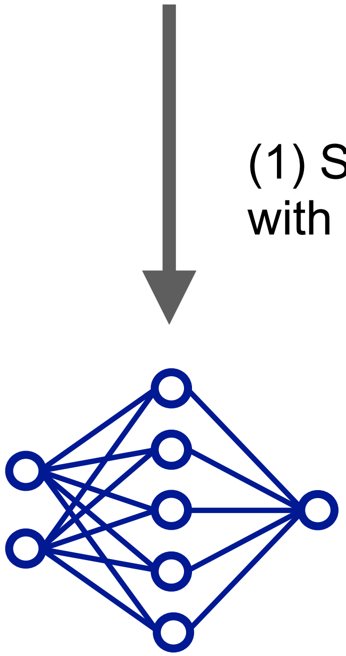


Particle ID

75% accurate

x 200 thousand

Fine-tuning



(1) Start with weights optimized with the larger dataset.

Fine-tuning / transfer learning

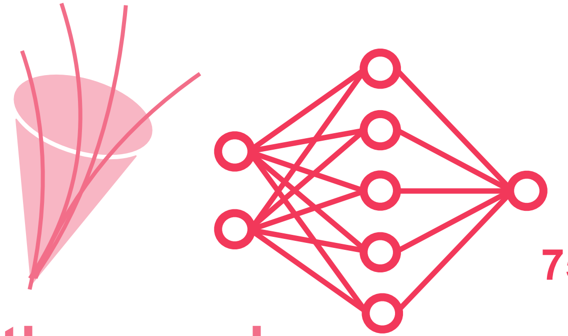
Original Task

Particle ID

95% accurate

x 20 million

Modified task



Particle ID

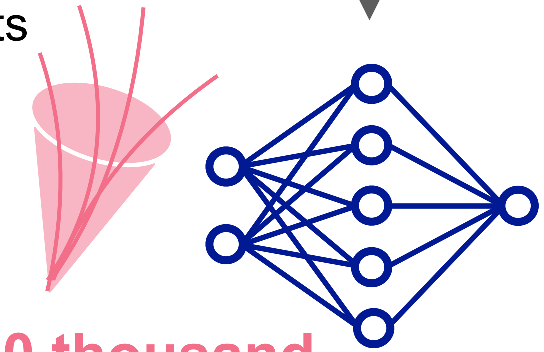
75% accurate 😣

x 200 thousand

Fine-tuning

(1) Start with weights optimized with the larger dataset.

(2) Adjust weights with the smaller dataset.



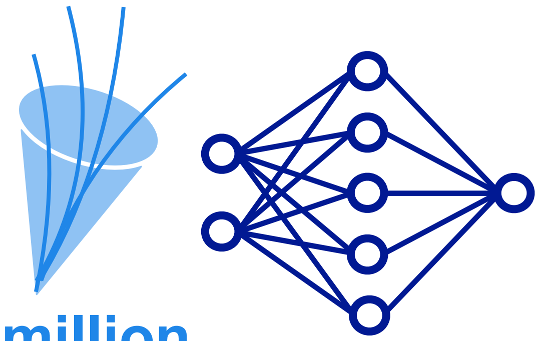
Particle ID

x 200 thousand

Actual example from my PhD!!

Fine-tuning / transfer learning

Original Task

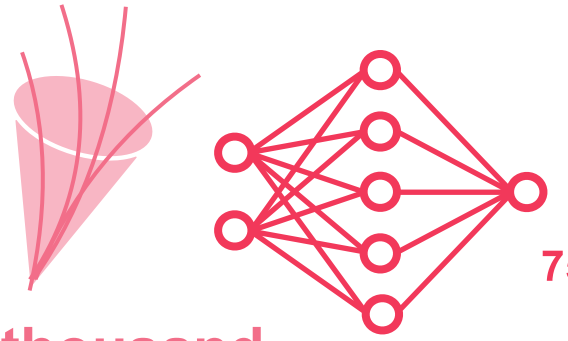


x 20 million

Particle ID

95% accurate

Modified task



x 200 thousand

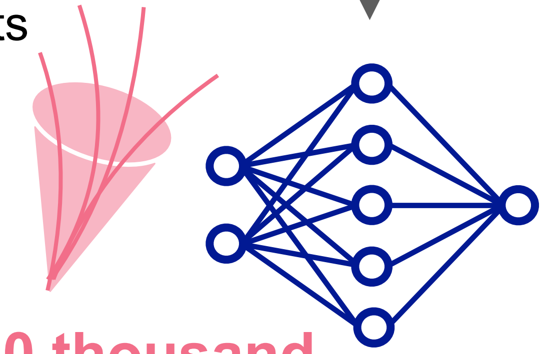
Particle ID

75% accurate

(1) Start with weights optimized with the larger dataset.

Fine-tuning

(2) Adjust weights with the smaller dataset.



x 200 thousand

Particle ID

95% accurate

(3) Enjoy increase in performance!

My conversation with chatGPT from Monday

Fine-tuning / Transfer learning: chatGPT

Original Task

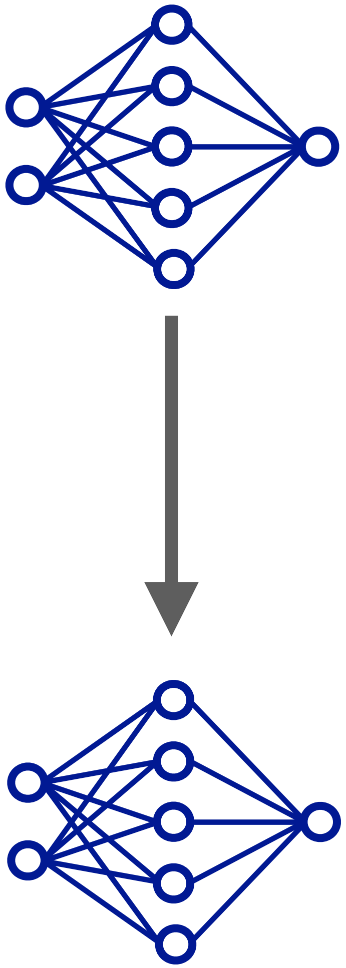
GPT-5

Train on the corpus of the whole internet

Fine-tuning

chatGPT

Fine-tune weights on the chatbot converations



$$F(y) = \int_0^y \frac{t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha, \beta)} dt = \sum_{i=\alpha}^{n} \binom{n}{i} y^i (1-y)^{n-i},$$

can you prove this relation

Thought for 1m 2s

Yes—this is the classic identity that the **regularized incomplete beta** equals a **binomial tail** when the shape parameters are positive integers.

Let $\alpha, \beta \in \mathbb{N}$ and set $n = \alpha + \beta - 1$. Define

$$F(y) = \int_0^y \frac{t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha, \beta)} dt, \qquad S(y) = \sum_{i=\alpha}^{n} \binom{n}{i} y^i (1-y)^{n-i}.$$

We will show $F(y) = S(y)$ for $y \in [0, 1]$.

Step 1: Differentiate the RHS

Differentiate term-by-term:

$$\frac{d}{dy}\left[y^i(1-y)^{n-i}\right] = i\, y^{i-1}(1-y)^{n-i} - (n-i)\, y^i (1-y)^{n-i-1}.$$

Using the binomial identities

$$i\binom{n}{i} = n\binom{n-1}{i-1}, \qquad (n-i)\binom{n}{i} = n\binom{n-1}{i},$$

we get

$$S'(y) = n\sum_{i=\alpha}^{n} \binom{n-1}{i-1} y^{i-1}(1-y)^{n-i} - n\sum_{i=\alpha}^{n} \binom{n-1}{i} y^i (1-y)^{n-i-1}.$$

Shift the index $j = i - 1$ in the first sum. The two sums then telescope, leaving only the lower boundary term:

$$S'(y) = n\binom{n-1}{\alpha-1} y^{\alpha-1}(1-y)^{n-\alpha}.$$

Since $n - \alpha = \beta - 1$,

$$S'(y) = n\binom{n-1}{\alpha-1} y^{\alpha-1}(1-y)^{\beta-1}.$$

Step 2: Match the normalizing constant

Compute

$$n\binom{n-1}{\alpha-1} = \frac{n!}{(\alpha-1)!(n-\alpha)!} = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!} = \frac{1}{B(\alpha, \beta)}.$$

Thus

$$S'(y) = \frac{y^{\alpha-1}(1-y)^{\beta-1}}{B(\alpha, \beta)}.$$

Step 3: Initial condition

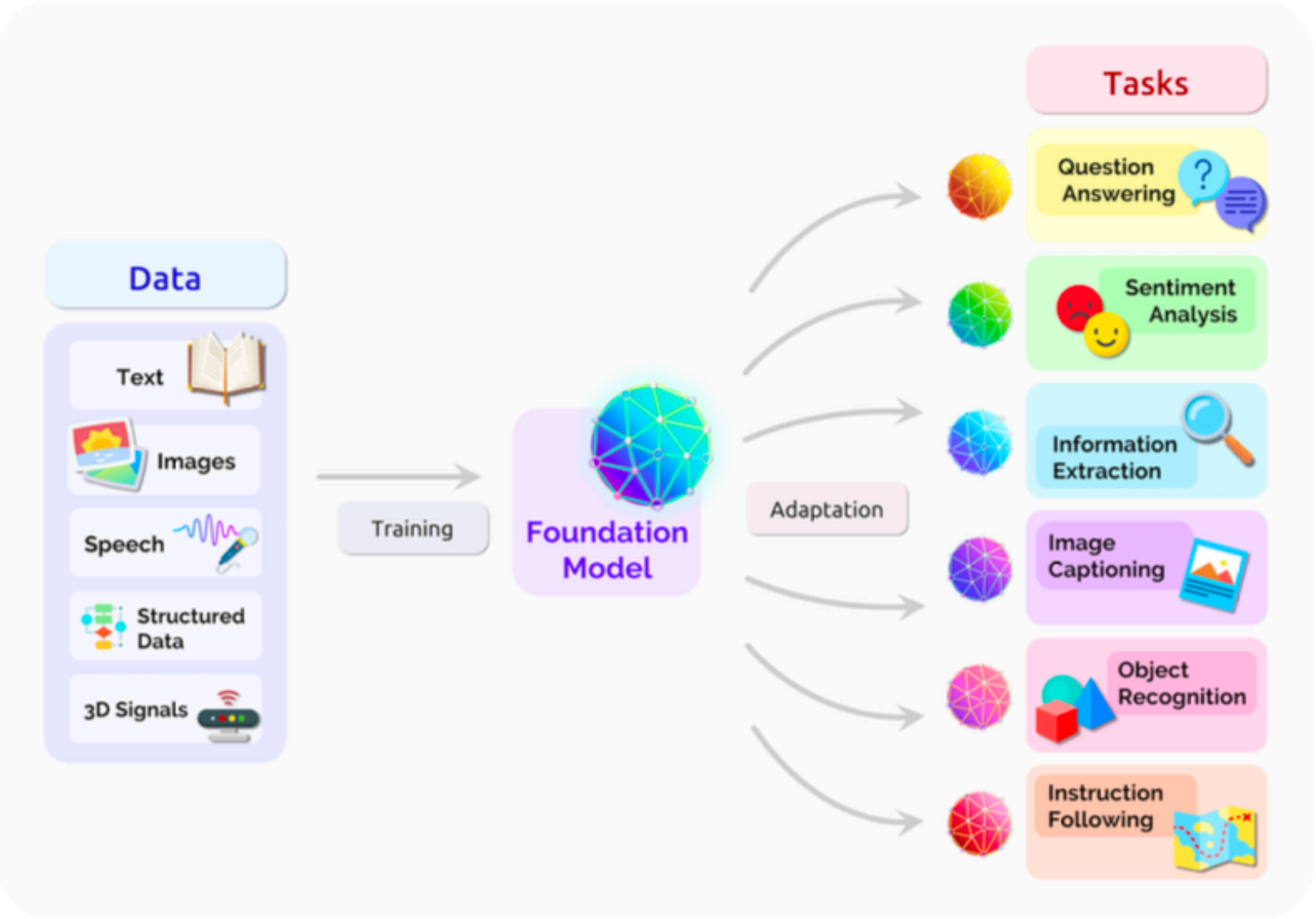
Both sides vanish at $y = 0$: $F(0) = 0$ by the integral limit, and $S(0) = 0$ because every term has a factor y^i with $i \geq \alpha \geq 1$.

Conclusion

Since $F'(y) = S'(y)$ on $[0, 1]$ and $F(0) = S(0)$, we have $F(y) = S(y)$ for all $y \in [0, 1]$:

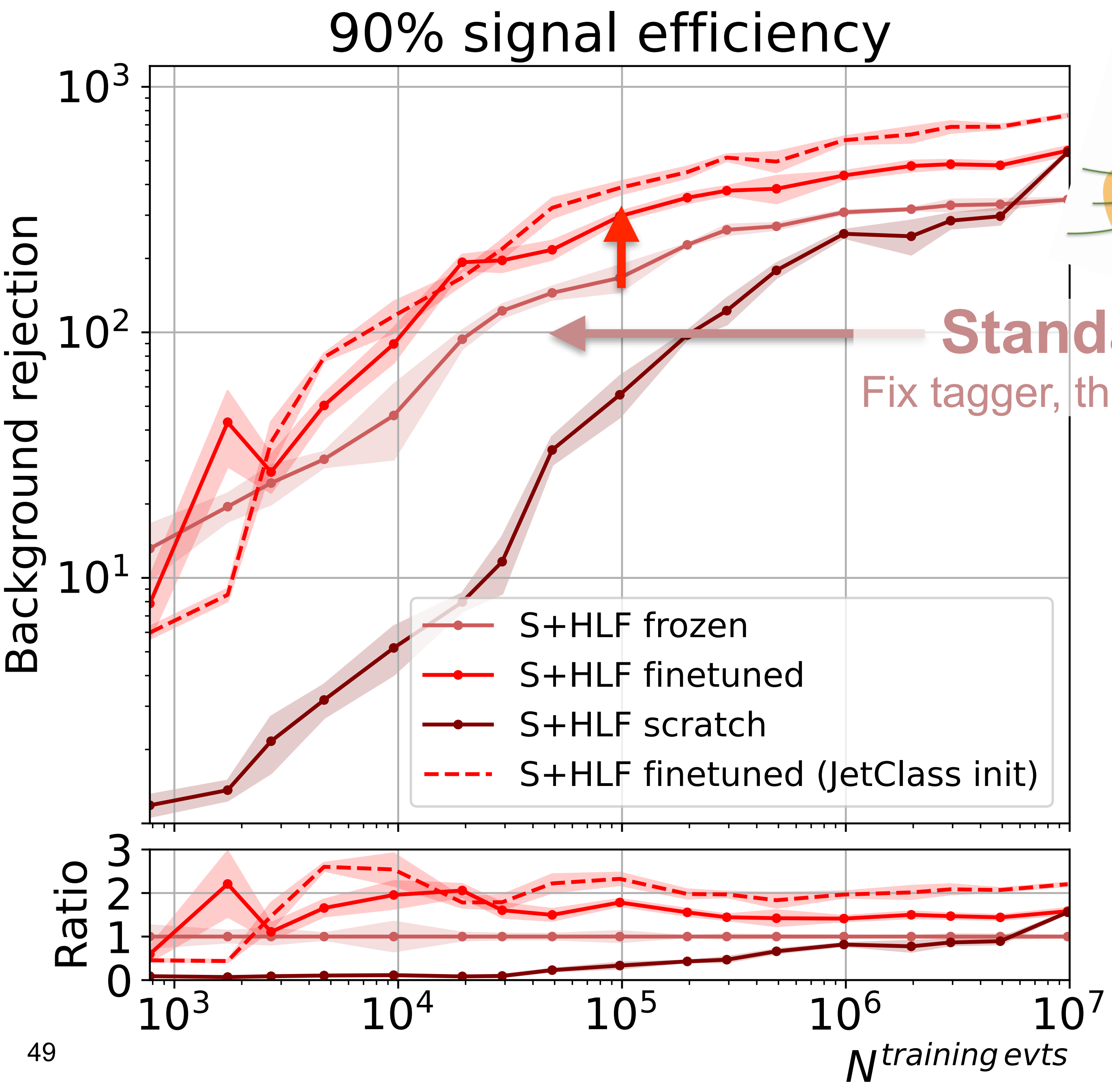
$$\boxed{\int_0^y \frac{t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha, \beta)} dt = \sum_{i=\alpha}^{n} \binom{n}{i} y^i (1-y)^{n-i}, \quad n = \alpha + \beta - 1}.$$

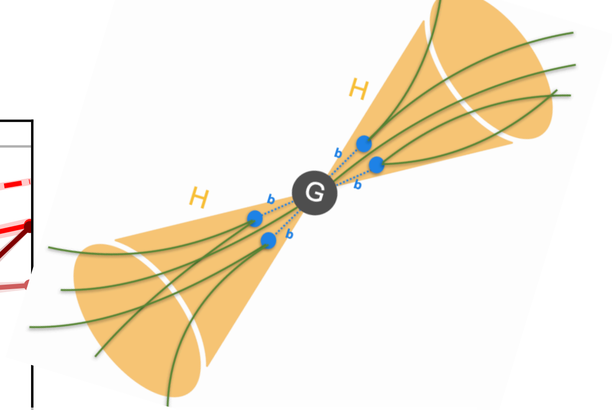
Interpretation: $F(y) = Iy(\alpha, \beta)$, the regularized incomplete beta. For integer α, β, it equals the **binomial tail** $\Pr\{X \geq \alpha\}$ for $X \sim \mathrm{Bin}(n, y)$ with $n = \alpha + \beta - 1$.



"A **foundation model** is any model that is trained on broad data (generally self-supervision at scale) that can be adapted (e.g, fine-tuned) to a wide range of downstream tasks."

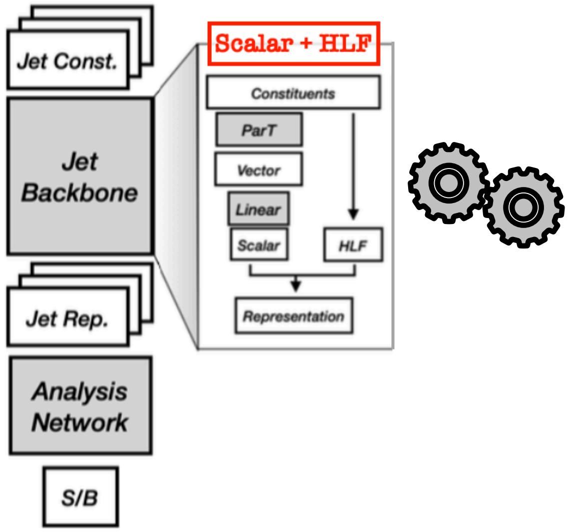
M. Vigl, **NH**, L. Heinrich
2401.13536





Standard HEP

Fix tagger, then train analysis



1.4 - 2x increase in background rejection with finetuning

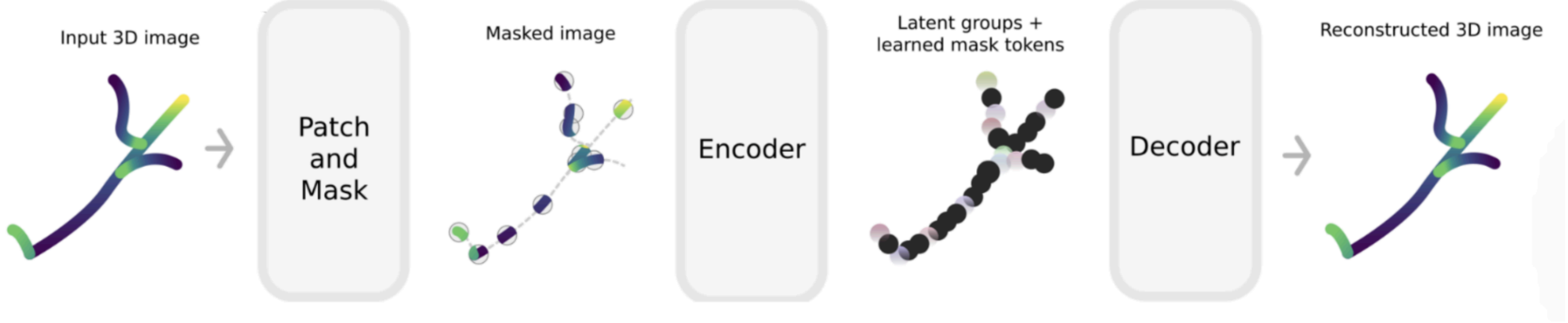
Stats limited analysis, $Z \sim S/\sqrt{B}$

↓ bkg by 2x same effect as ↑ $\mathscr{L}$ by 2x

Neutrino physics (DUNE)

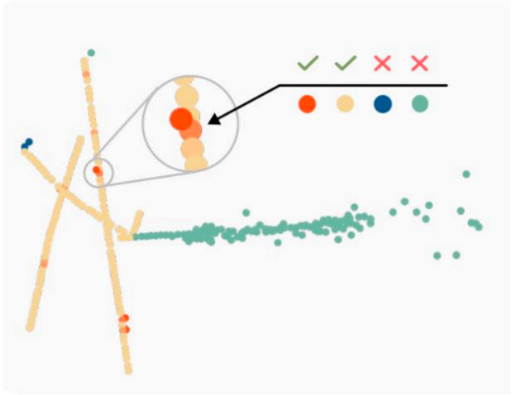
K. Terao slide
2502.02558

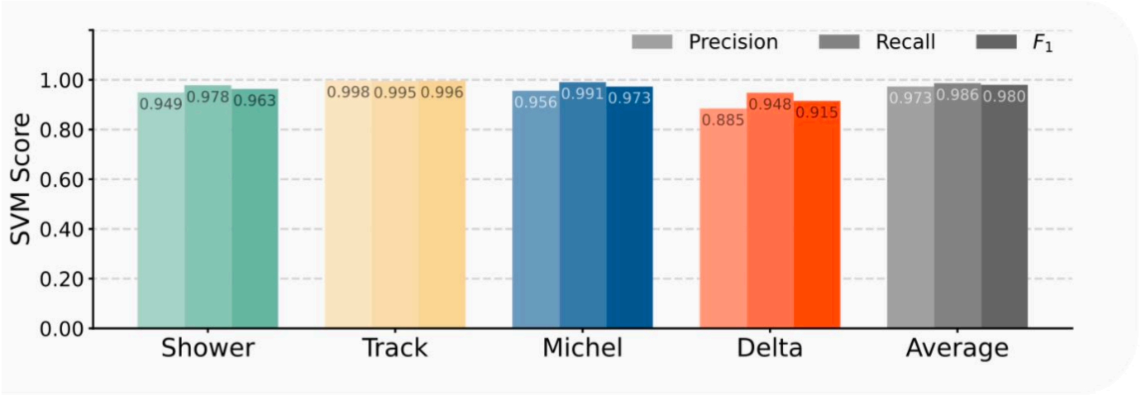
Pre-training: unlabelled data



Maybe don't even need to finetune! " 0 shot tranfer"

Linear probing





The Q: Build big or build smart?

BUILD BIG OR BUILD SMART: EXAMING SCALE AND DOMAIN KNOWLEDGE IN MACHINE LEARNING FOR FUNDAMENTAL PHYSICS

25 August - 19 September 2025

Lukas Heinrich, Michael Kagan, Margarita Osadchy, Tobias Golling, Siddarth Mishra-Sharma

Workshop ongoing in Munich, 25.8 — 19.9

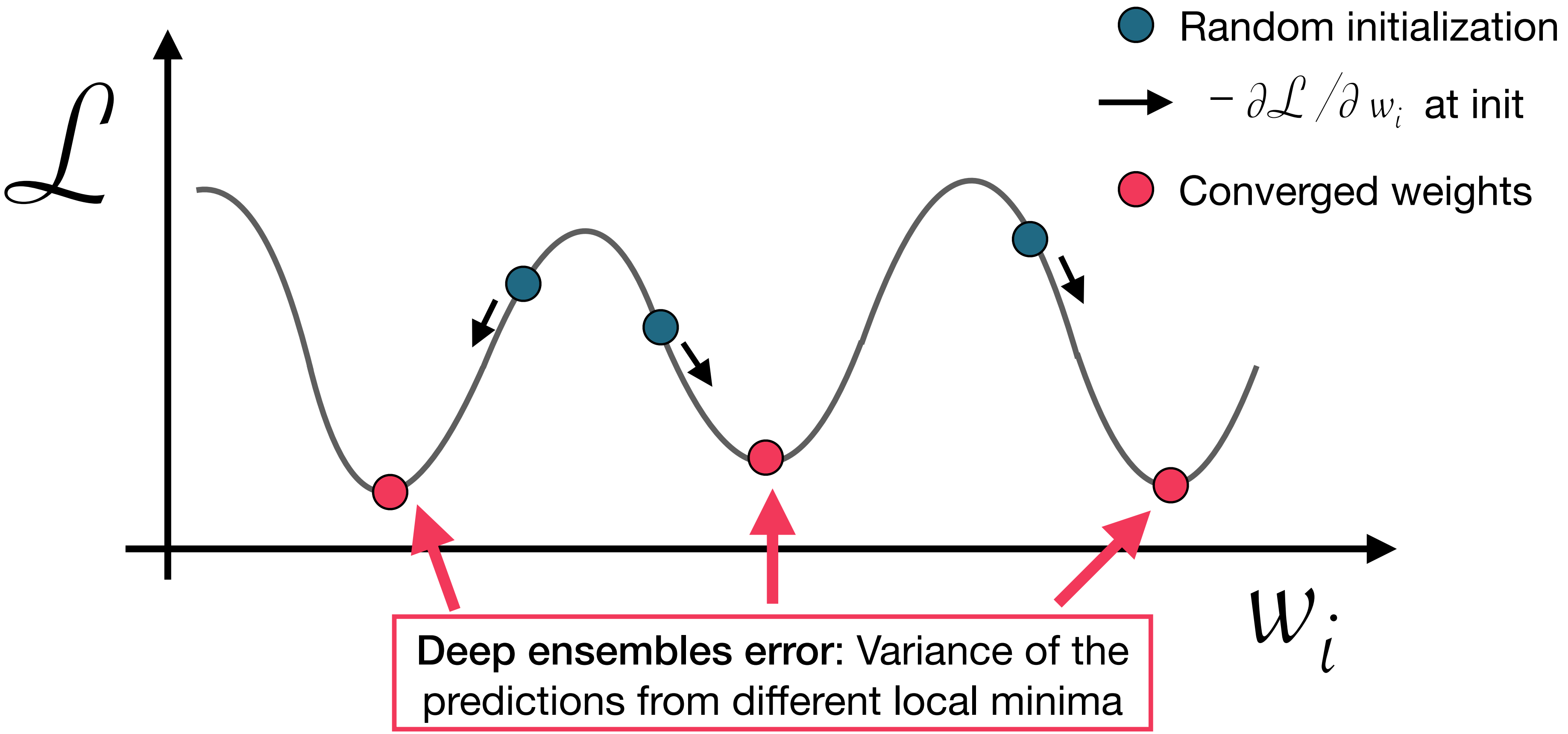
Mini ws on foundation models: https://indico.ph.tum.de/event/7906/timetable/

Month-long program: https://www.munich-iapbp.de/activities/activities-2025/machine-learning

Ensembles: how to quantify the error on your model

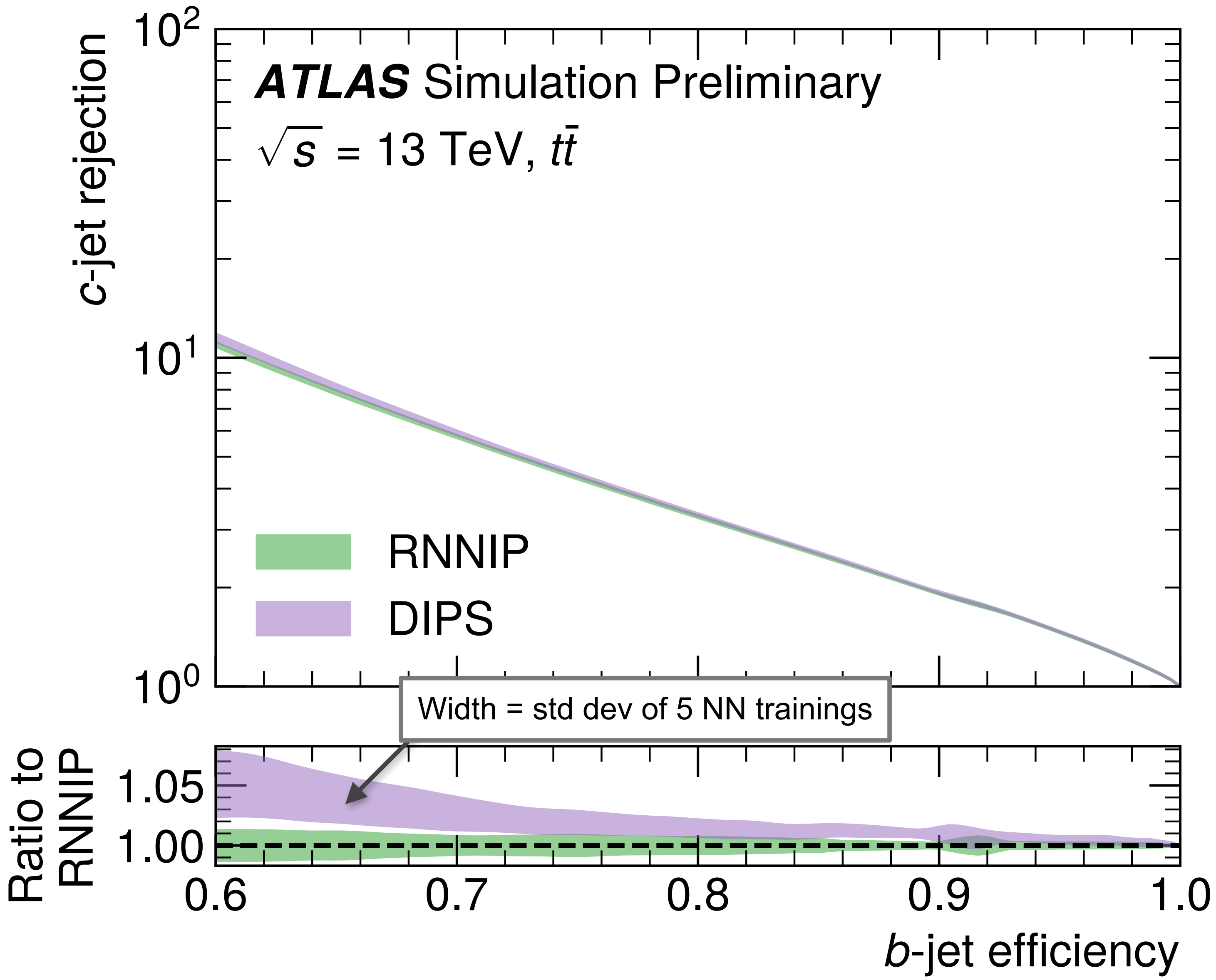
$$MSE(x) \approx (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

Variance

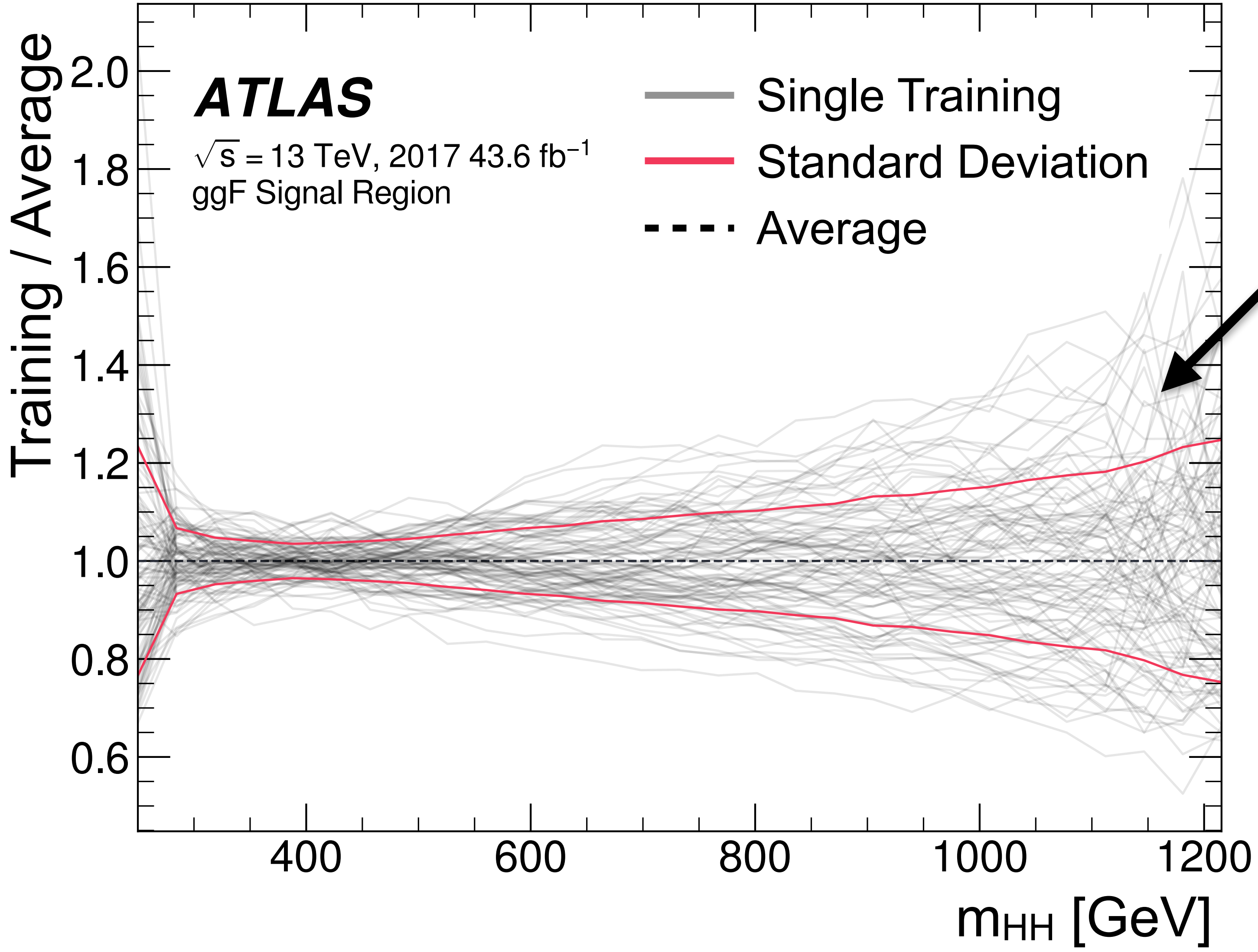


Ensembles: Application

Probe whether the result of an experiment is meaningful or a random fluctuation.



Ensembles: Application



Use the variation of trainings as a nuisance parameter

Starting off...

Open question

Going deeper

Training techniques

Statistical learning theory

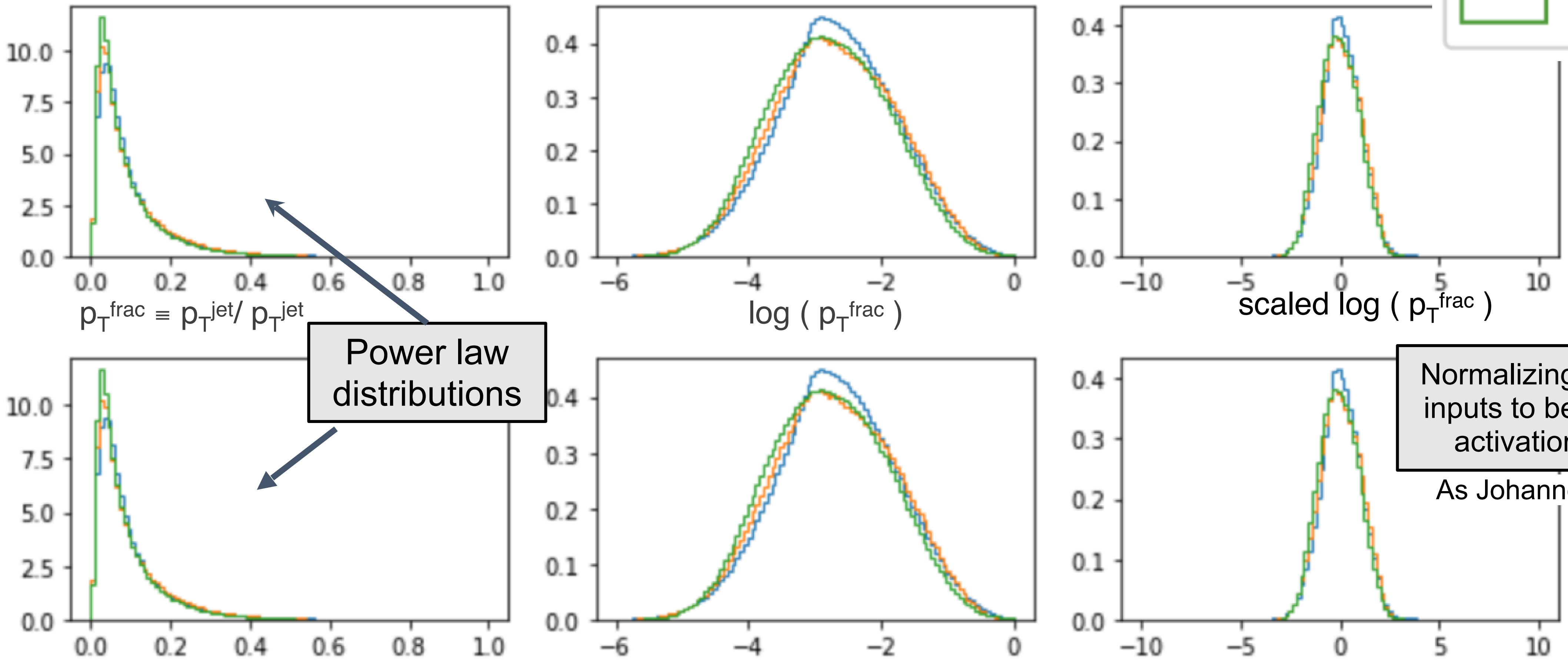
Bias / variance trade-off

✋ Would love feedback + discussions!

Log transform: motivation

Ex: b-tagging input features

🧐 Somewhat HEP specific which often have features with these falling spectra



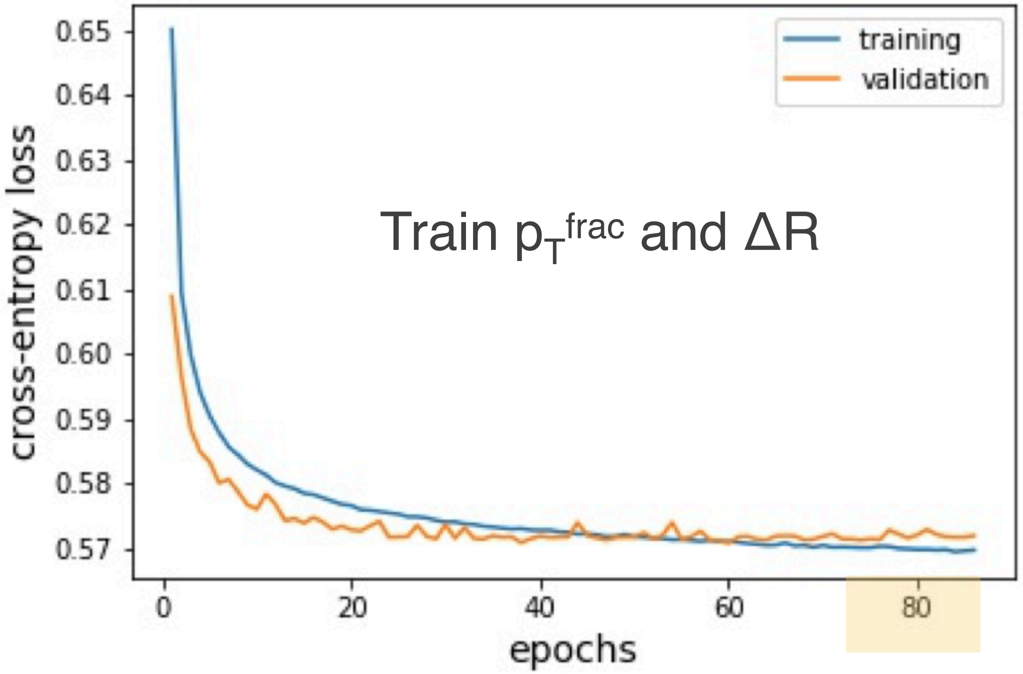
Power law distributions

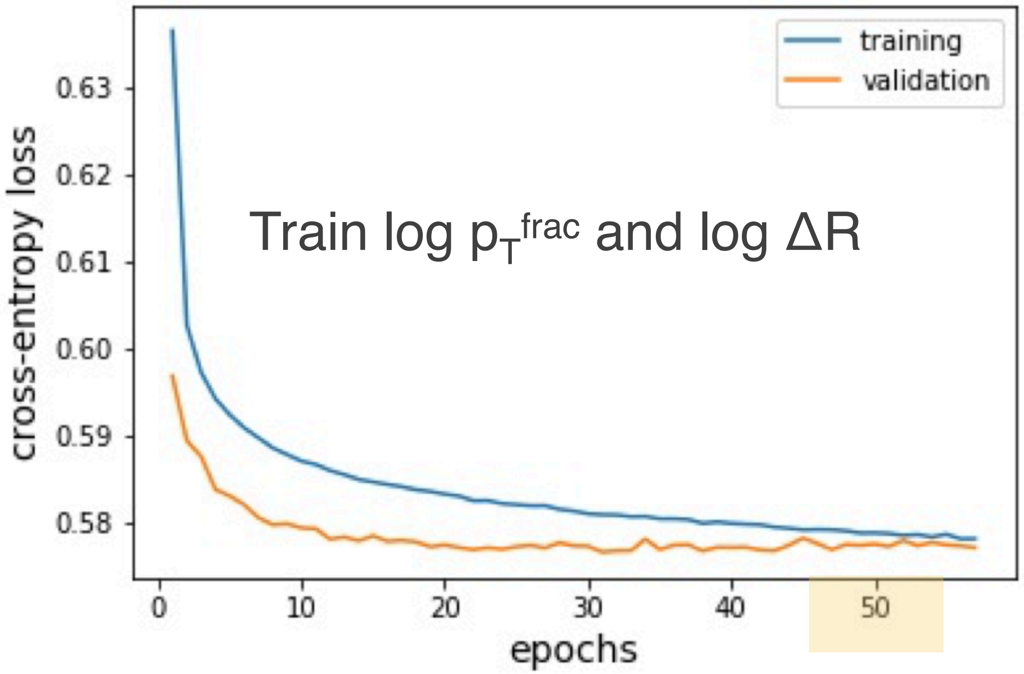
Normalizing encourages inputs to be close to the activation functions

As Johannes explained

With the log, become bell-shape

Log transform: motivation

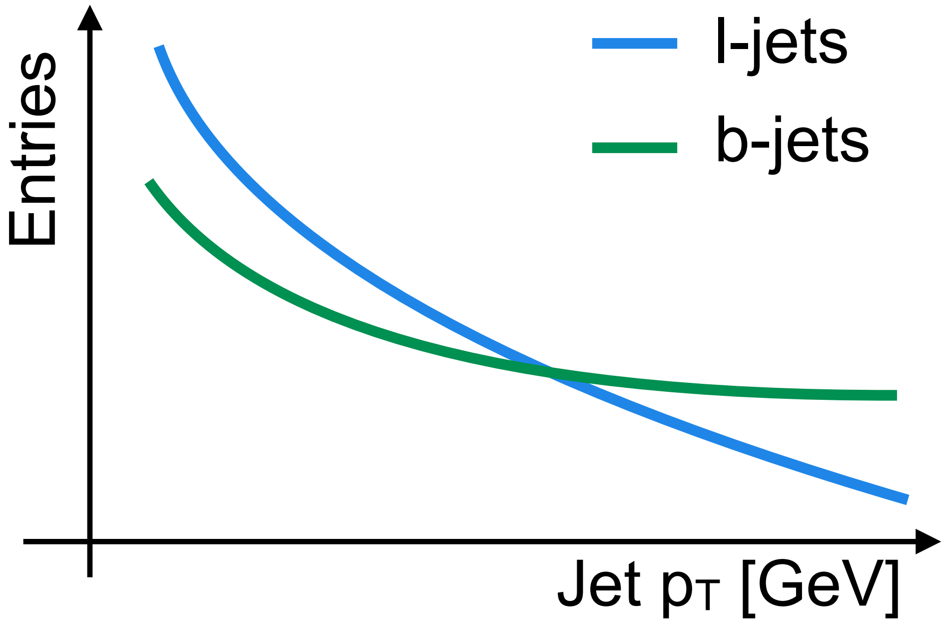




How does this help? 20% speed up in training time!

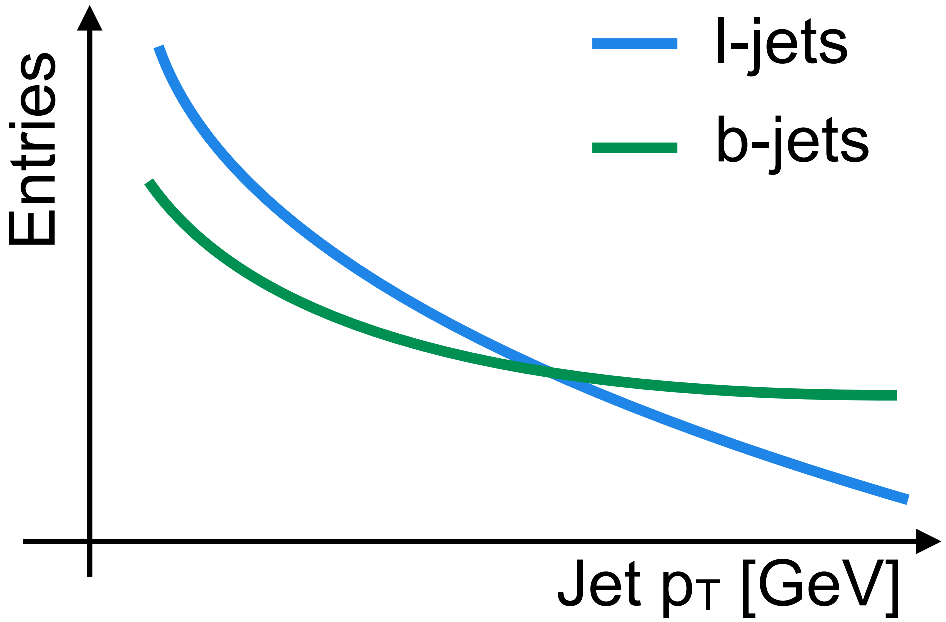
Sample dependence

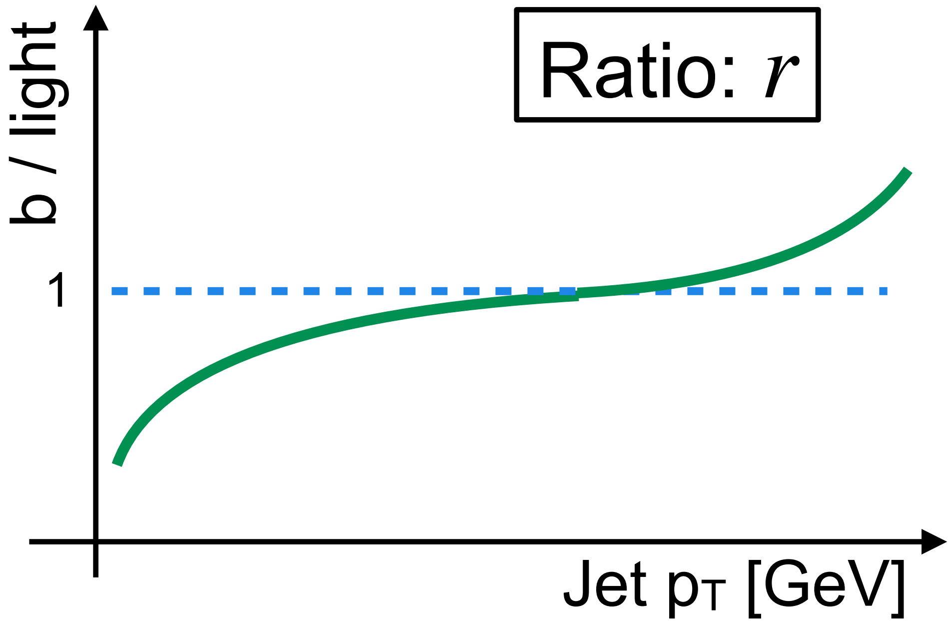
Issue: Want a classifier that is performant over a range of energies



Sample dependence

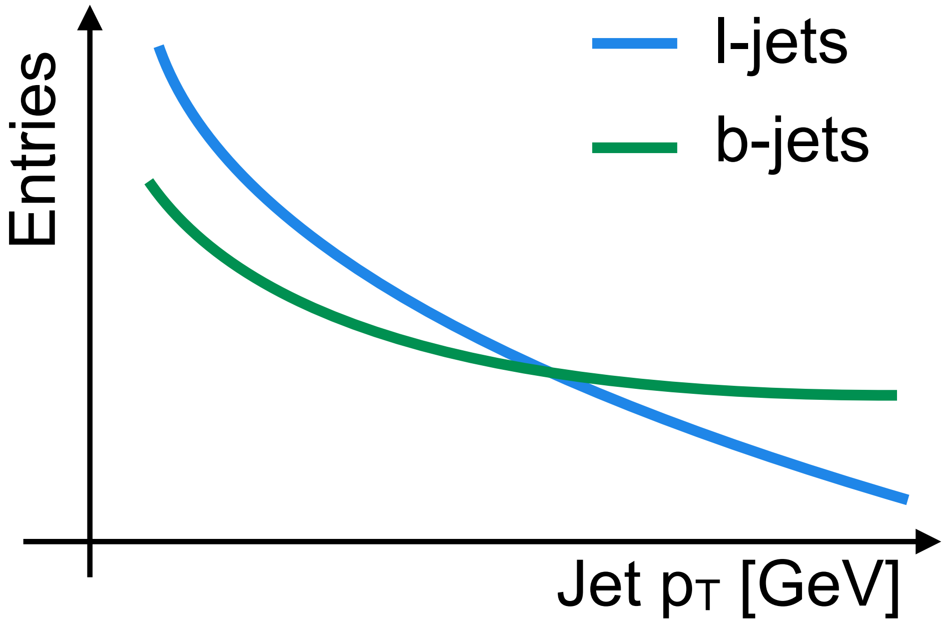
Issue: Want a classifier that is performant over a range of energies

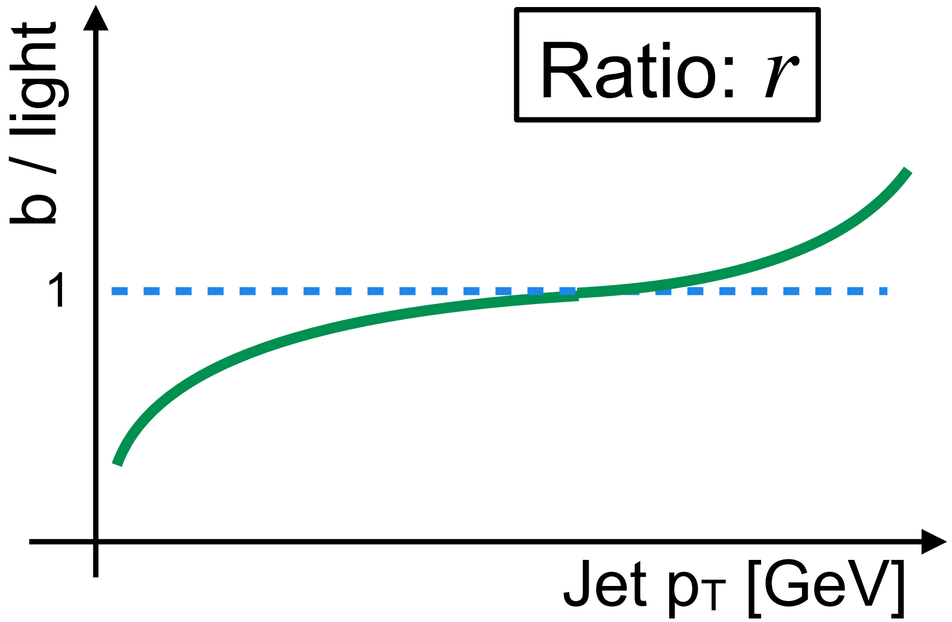




Sample dependence

Issue: Want a classifier that is performant over a range of energies





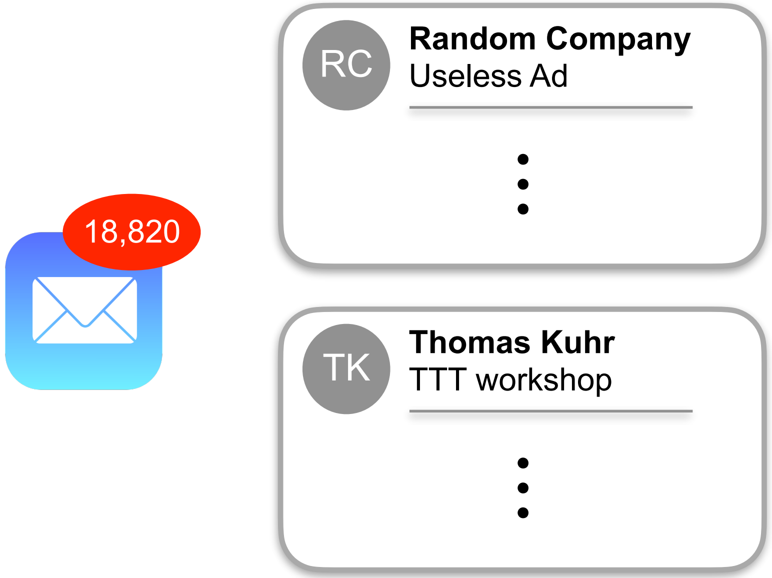
$$w_i = w_i - \alpha \sum_{j=1}^{M} \frac{1}{r(p_T^{(j)})} \nabla_w \mathscr{L}_j$$

sample weight

- l-jets: $r = 1$
- b-jets: r = b/l ratio

Ablation studies: What has the model learned?

Example — spam classification



❌ **Spam**

✅ **Not Spam**

baseline:
94.0% accuracy

Ablation studies: What has the model learned?

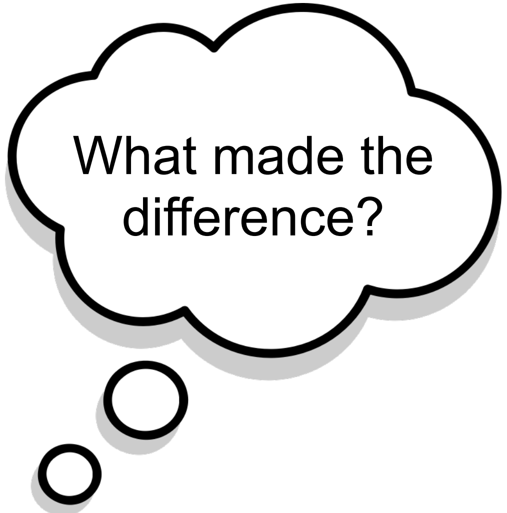
Example — spam classification

RC **Random Company**
Useless Ad
⋮

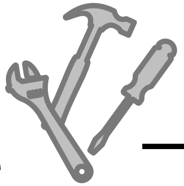
❌ **Spam**

TK **Thomas Kuhr**
TTT workshop
⋮

✅ **Not Spam**



baseline:
94.0% accuracy

Feature engineering  → NN → ❌ **Spam** / ✅ **Not Spam**

Overall system:
99.9% accuracy

Ablation studies

Remove features from the model…
and see what breaks it!

Component	Accuracy
Overall system	99.9%
Spelling correction	99.0
Sender host features	98.9%
Email header features	98.9%
Email text parser features	95%
Javascript parser	94.5%
Features from images	94.0%

Email text parser: most important feature!

[baseline]

Debugging exercise

[If ⏰ permits]

Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

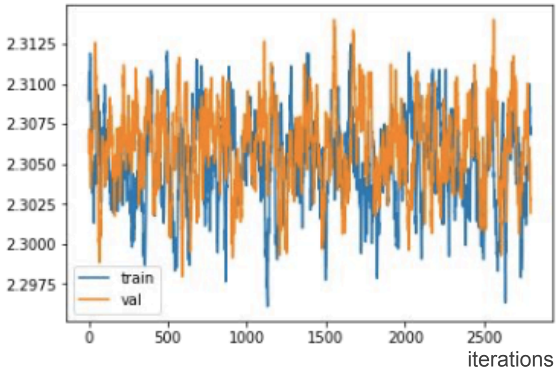
Applied the negative of the gradients

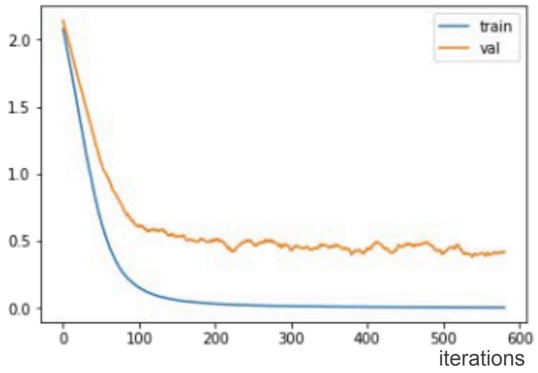
Not converged yet: need longer training

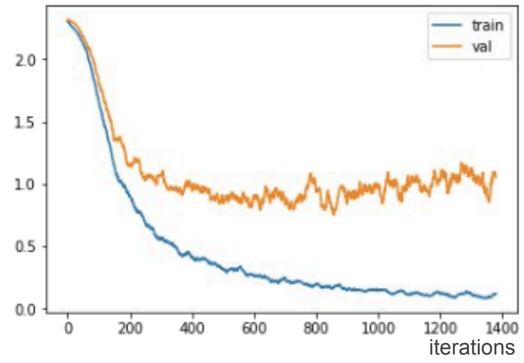
Not learning: gradients not applied to the weights

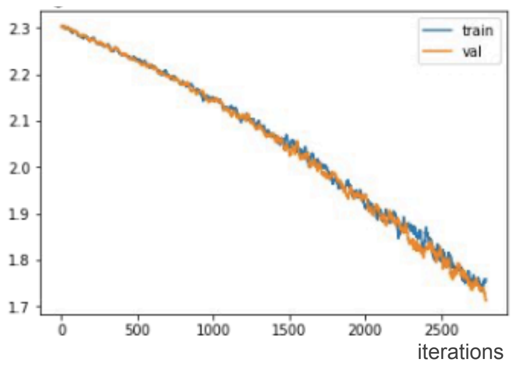
Overfit: model too large / dataset too small

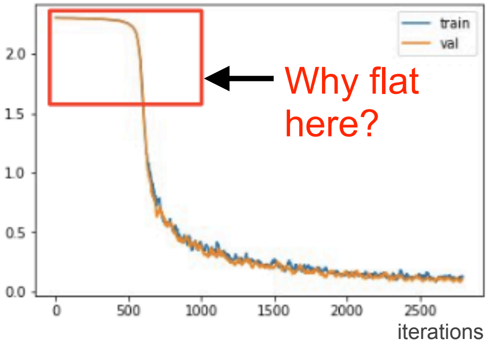
Loss curves

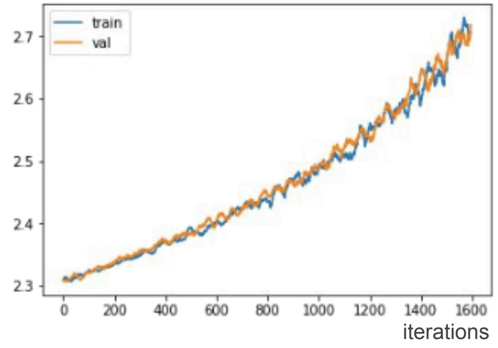












Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

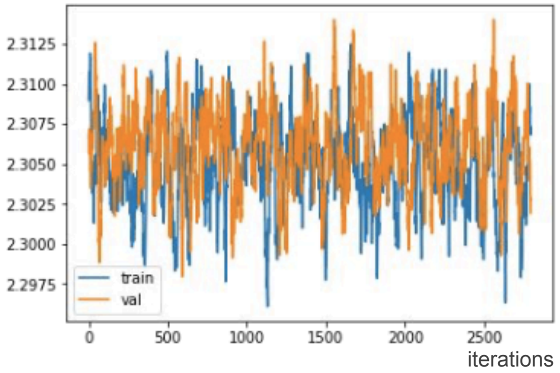
Applied the negative of the gradients

Not converged yet: need longer training

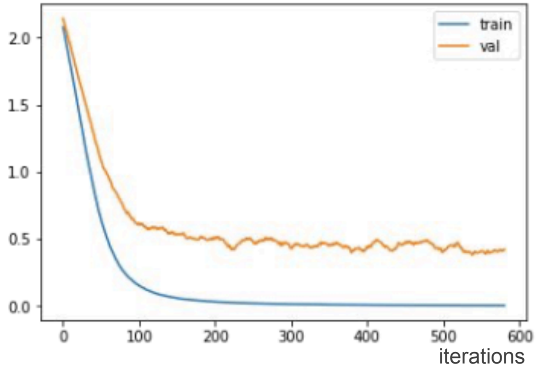
Not learning: gradients not applied to the weights

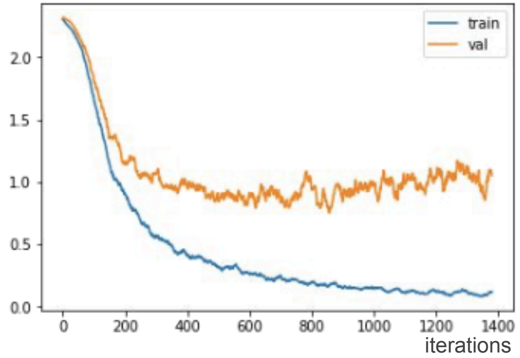
Overfit: model too large / dataset too small

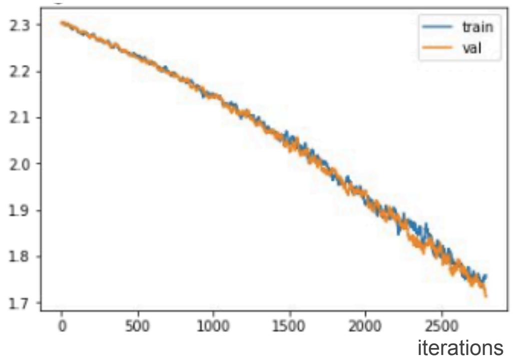
Loss curves

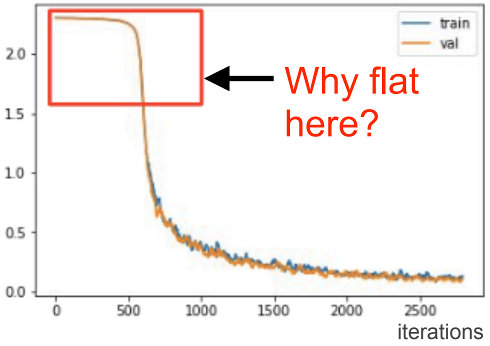


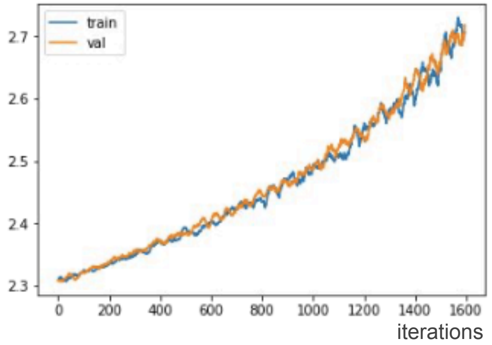
Not learning: gradients not applied to the weights











Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

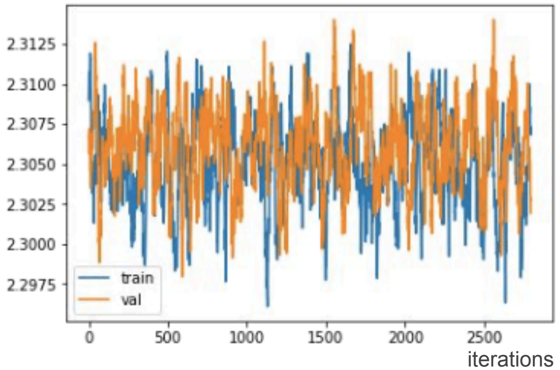
Applied the negative of the gradients

Not converged yet: need longer training

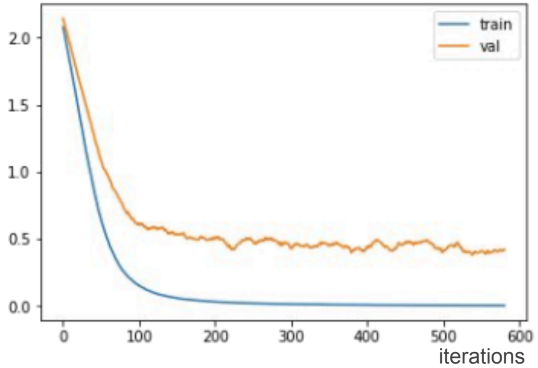
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

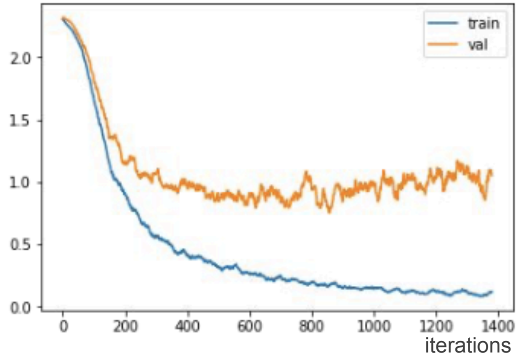
Loss curves

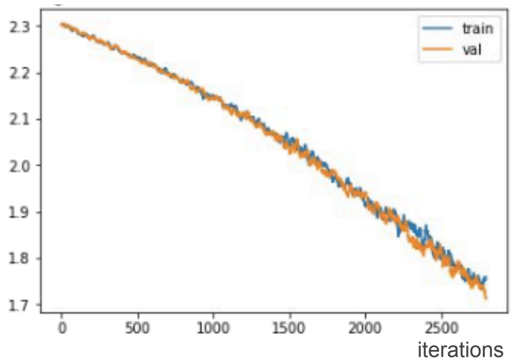


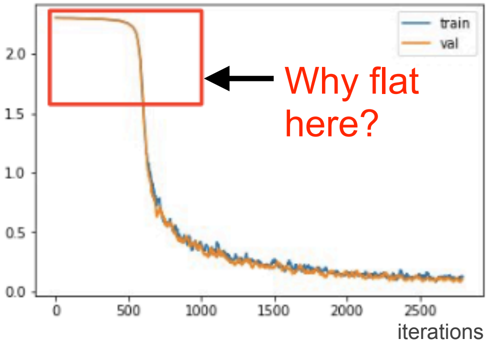
Not learning: gradients not applied to the weights

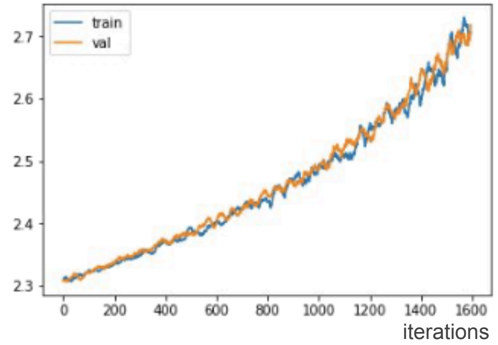


Overfit: model too large / dataset too small









Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

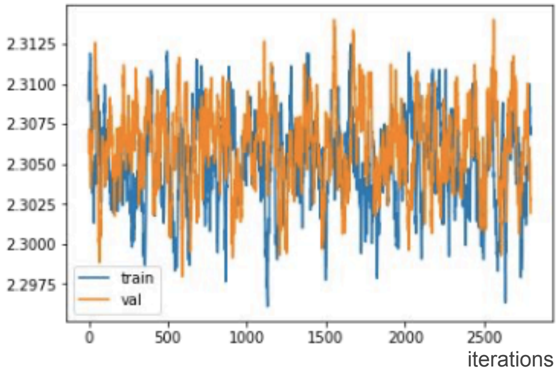
Applied the negative of the gradients

Not converged yet: need longer training

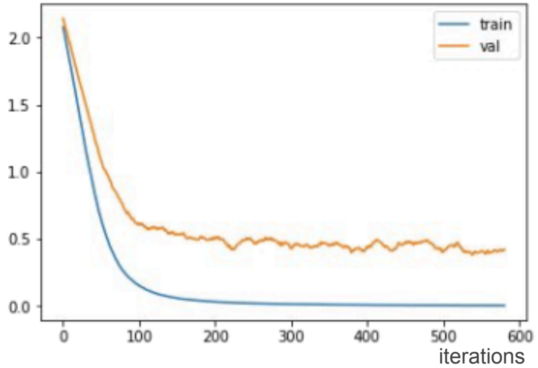
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

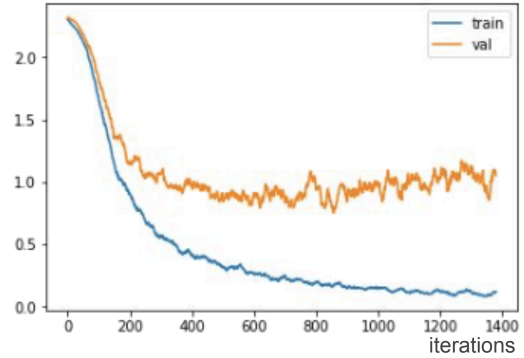
Loss curves



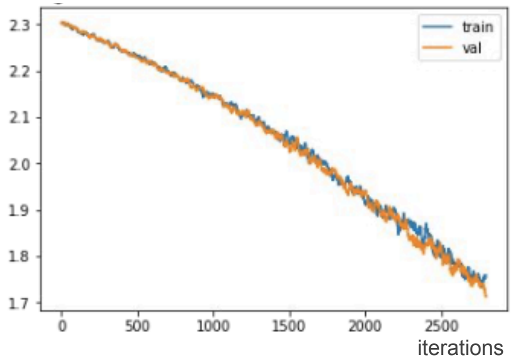
Not learning: gradients not applied to the weights

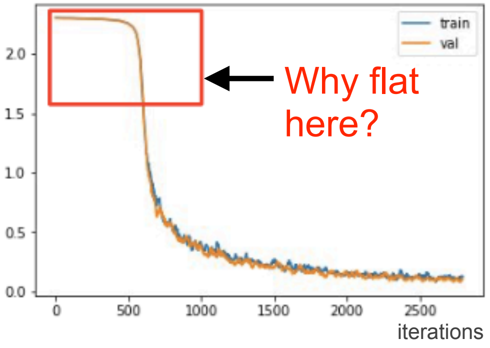


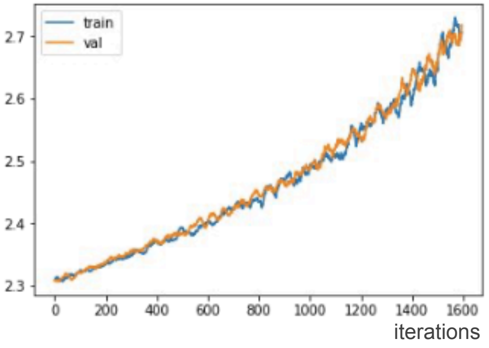
Overfit: model too large / dataset too small



More extreme case of overfitting







Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

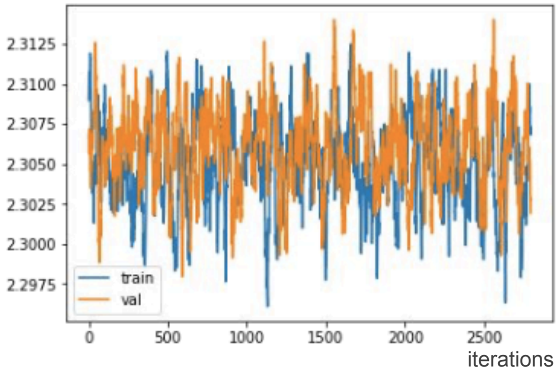
Applied the negative of the gradients

Not converged yet: need longer training

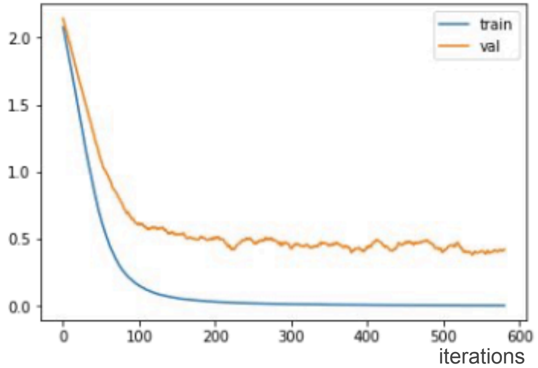
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

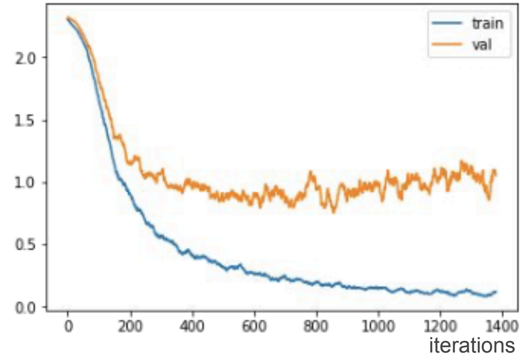
Loss curves



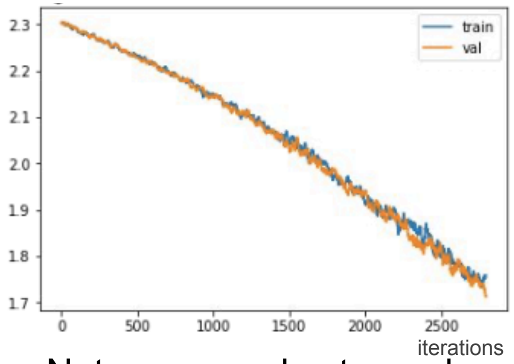
Not learning: gradients not applied to the weights



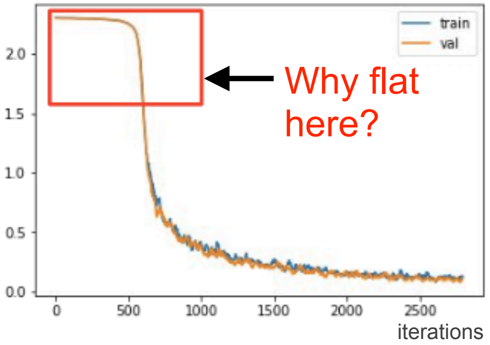
Overfit: model too large / dataset too small

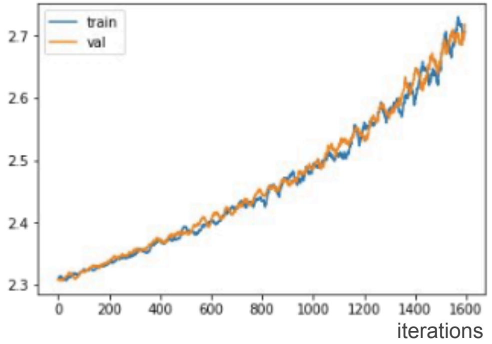


More extreme case of overfitting



Not converged yet: need longer training





Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

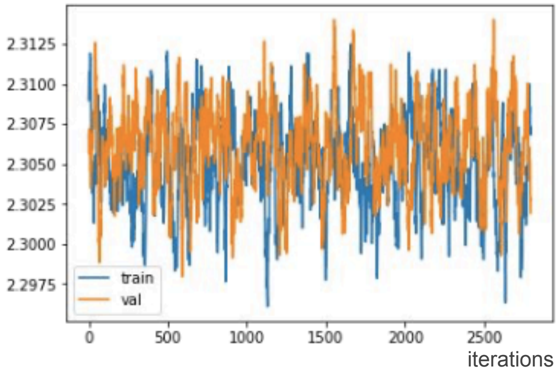
Applied the negative of the gradients

Not converged yet: need longer training

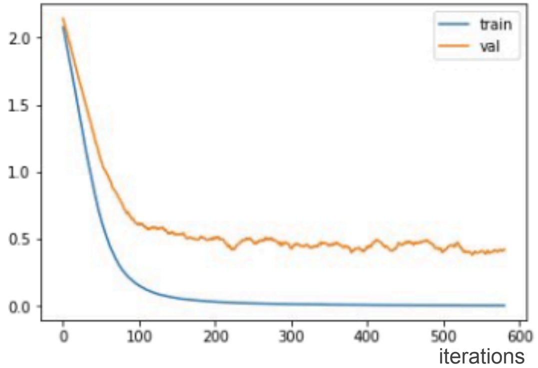
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

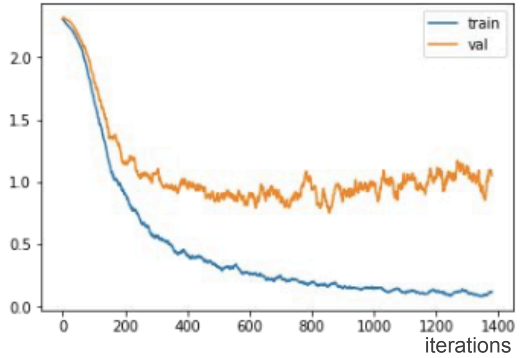
Loss curves



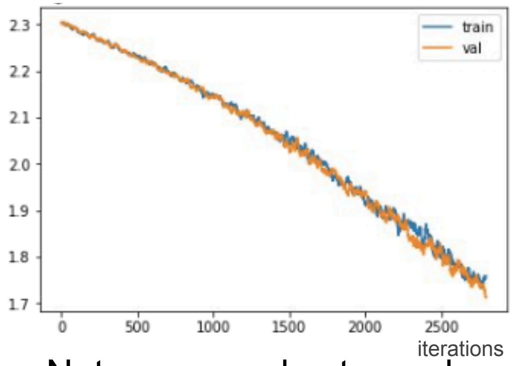
Not learning: gradients not applied to the weights



Overfit: model too large / dataset too small

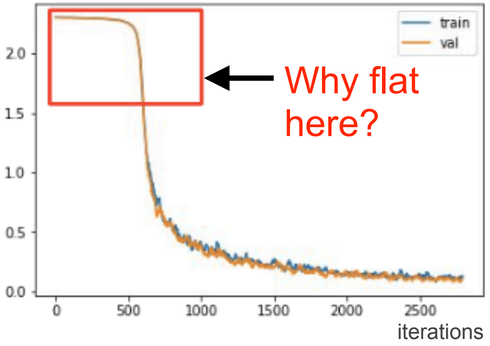


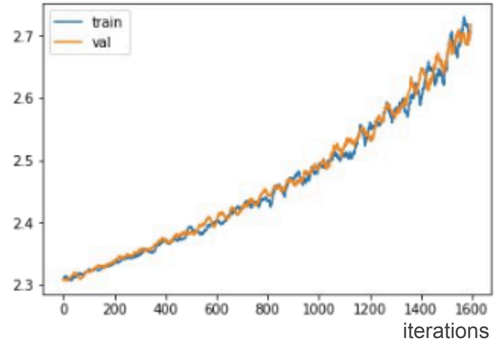
More extreme case of overfitting



Not converged yet: need longer training

Slow start





Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

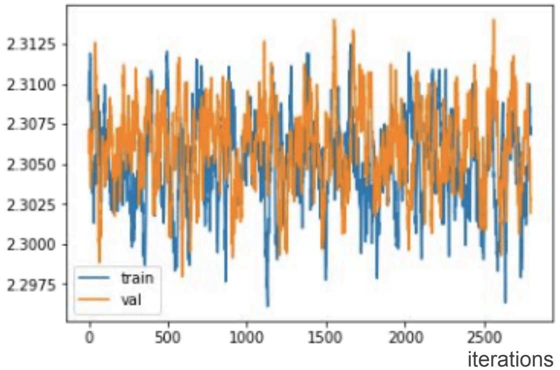
Applied the negative of the gradients

Not converged yet: need longer training

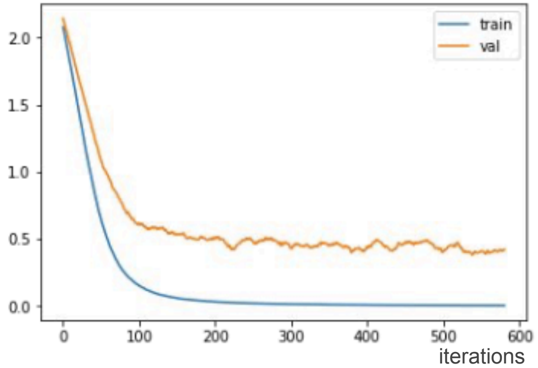
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

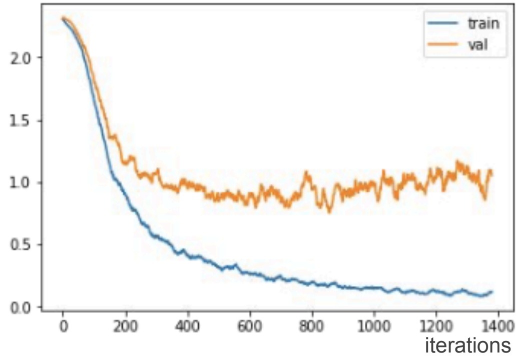
Loss curves



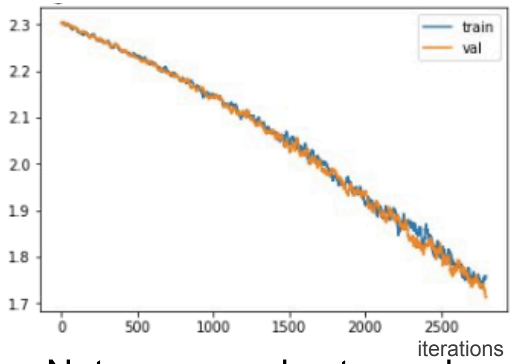
Not learning: gradients not applied to the weights



Overfit: model too large / dataset too small

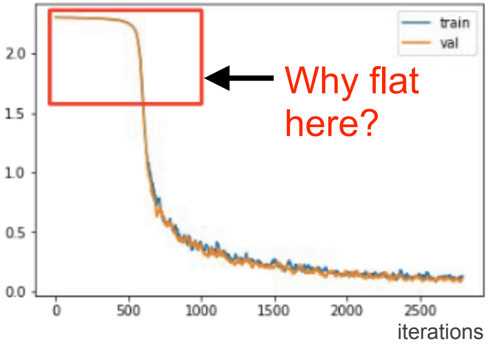


More extreme case of overfitting

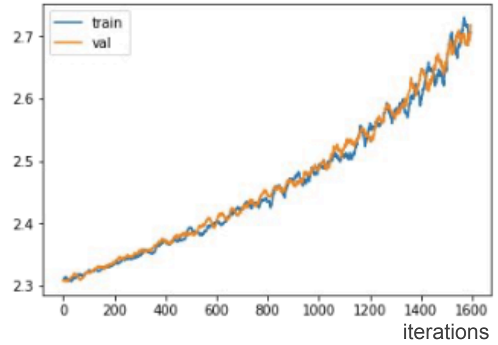


Not converged yet: need longer training

Slow start



Slow start: initialization learning rate too small



Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

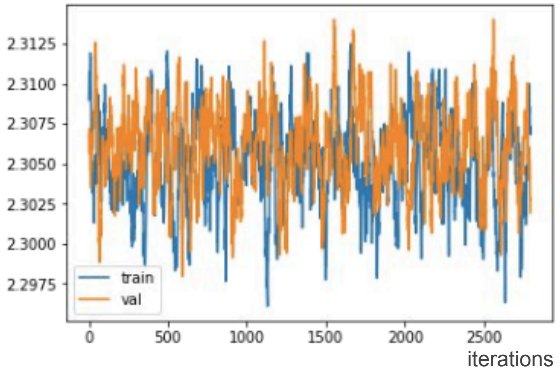
Applied the negative of the gradients

Not converged yet: need longer training

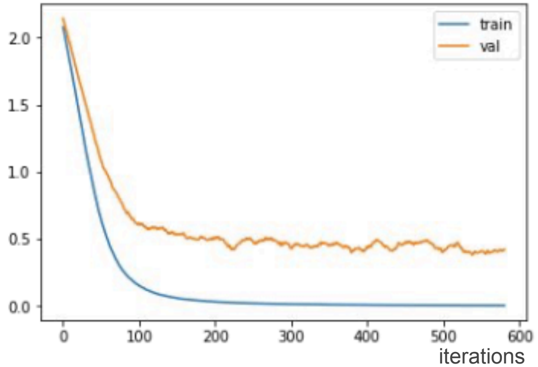
Not learning: gradients not applied to the weights

Overfit: model too large / dataset too small

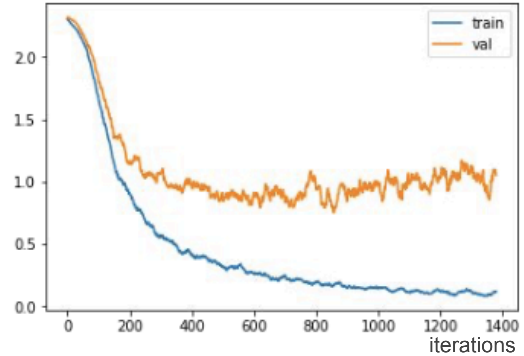
Loss curves



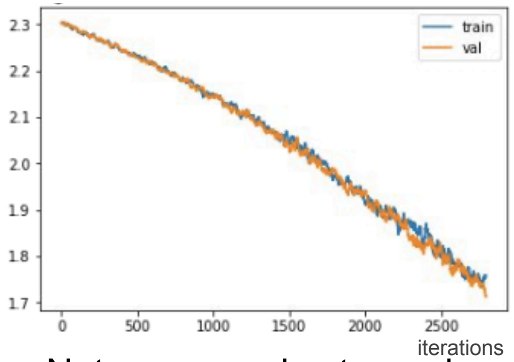
Not learning: gradients not applied to the weights



Overfit: model too large / dataset too small

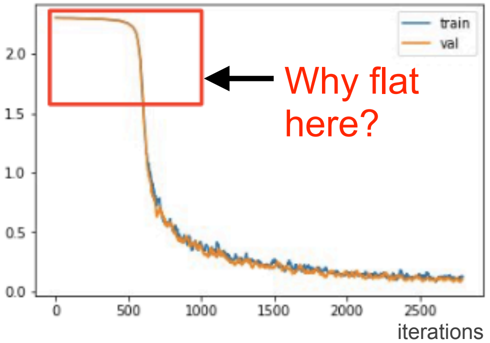


More extreme case of overfitting

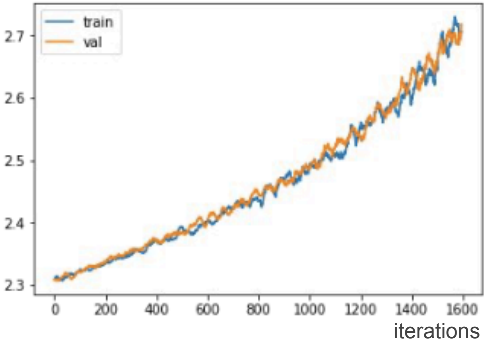


Not converged yet: need longer training

Slow start



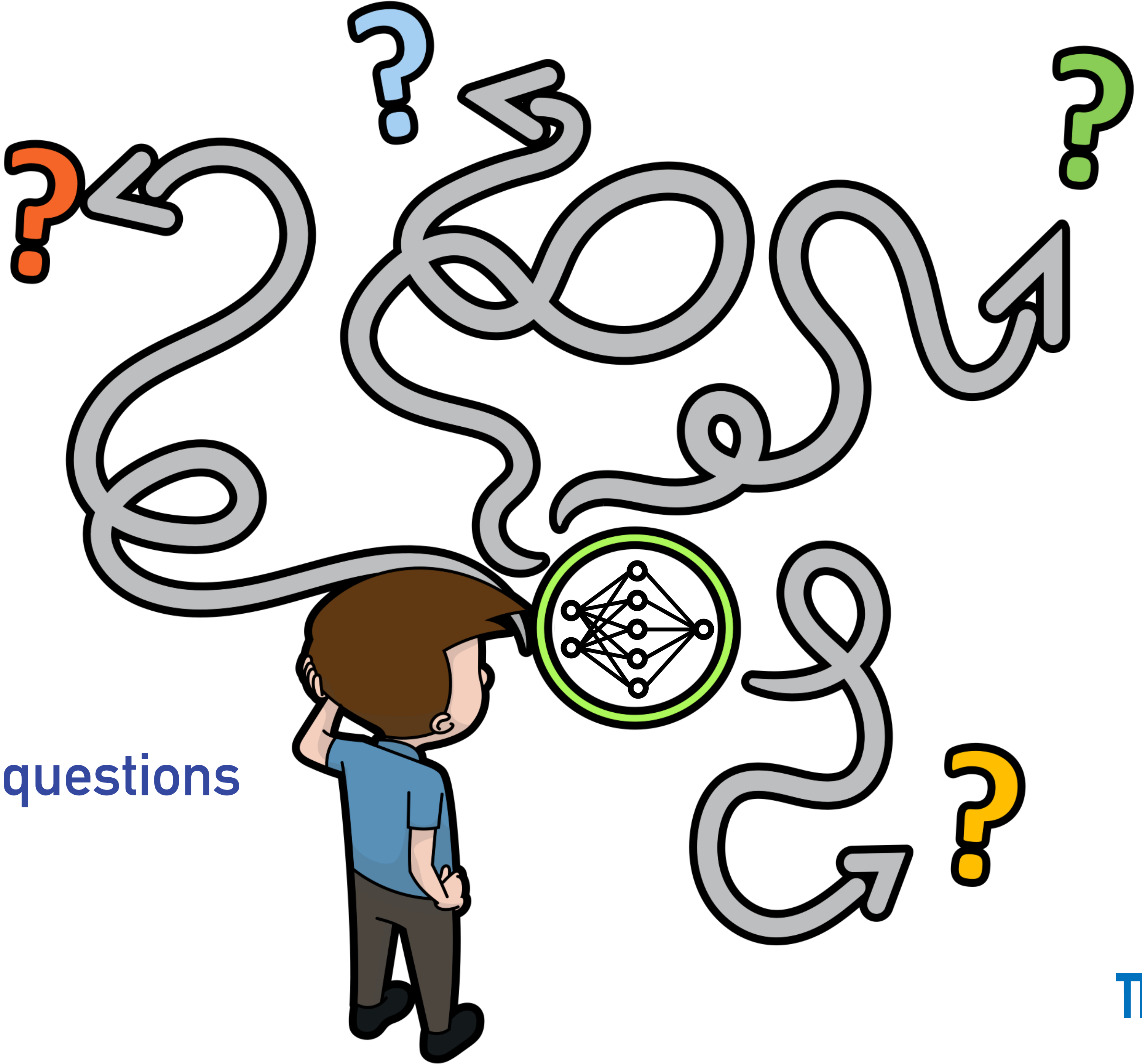
Slow start: initialization learning rate too small



Applied the negative of the gradients

Starting off...

Would love feedback + discussions!



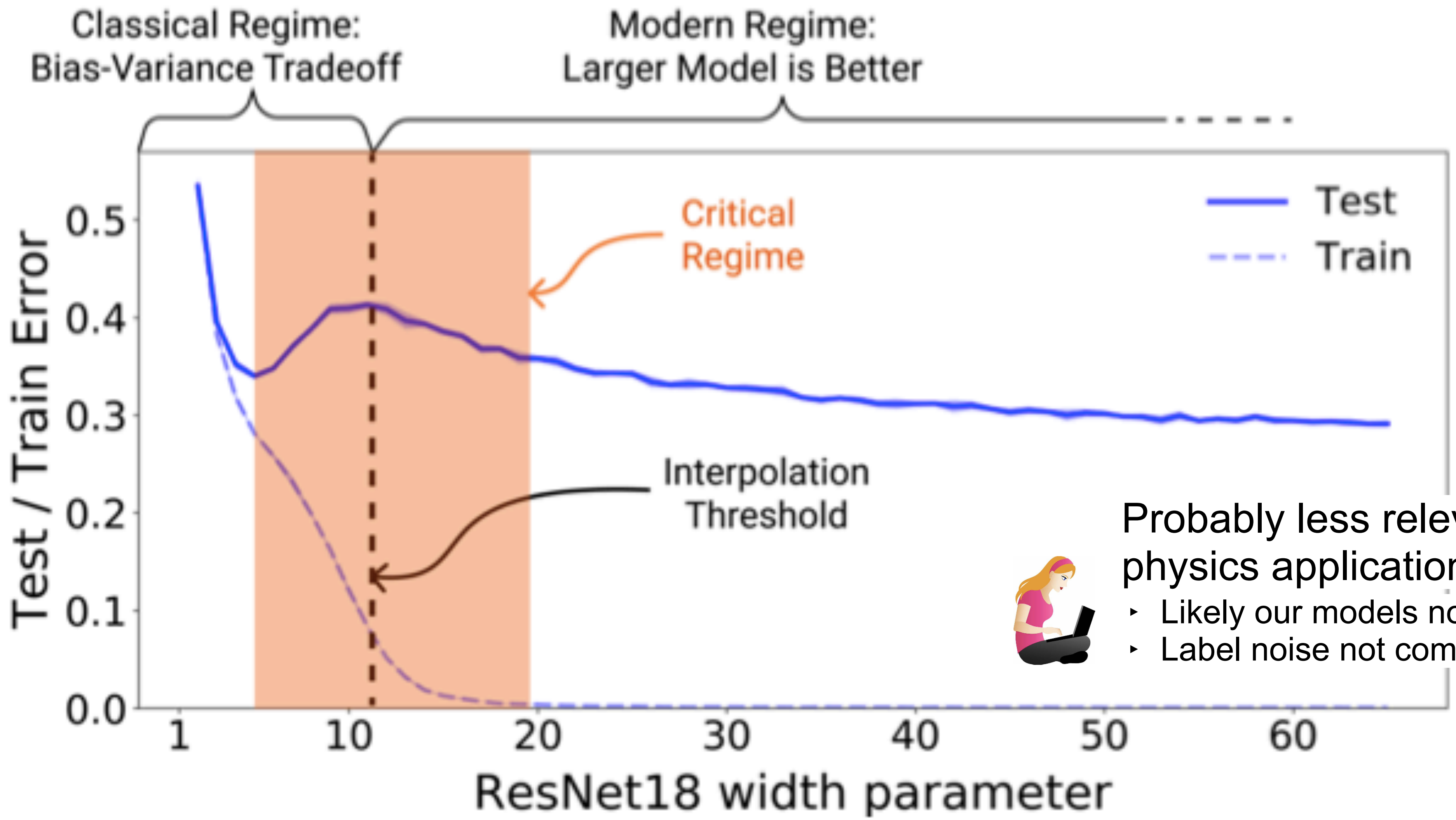
[One of the]

Outstanding deep questions in deep learning.

Double Descent

1912.02292

Daniela Witten
@daniela_witten
post



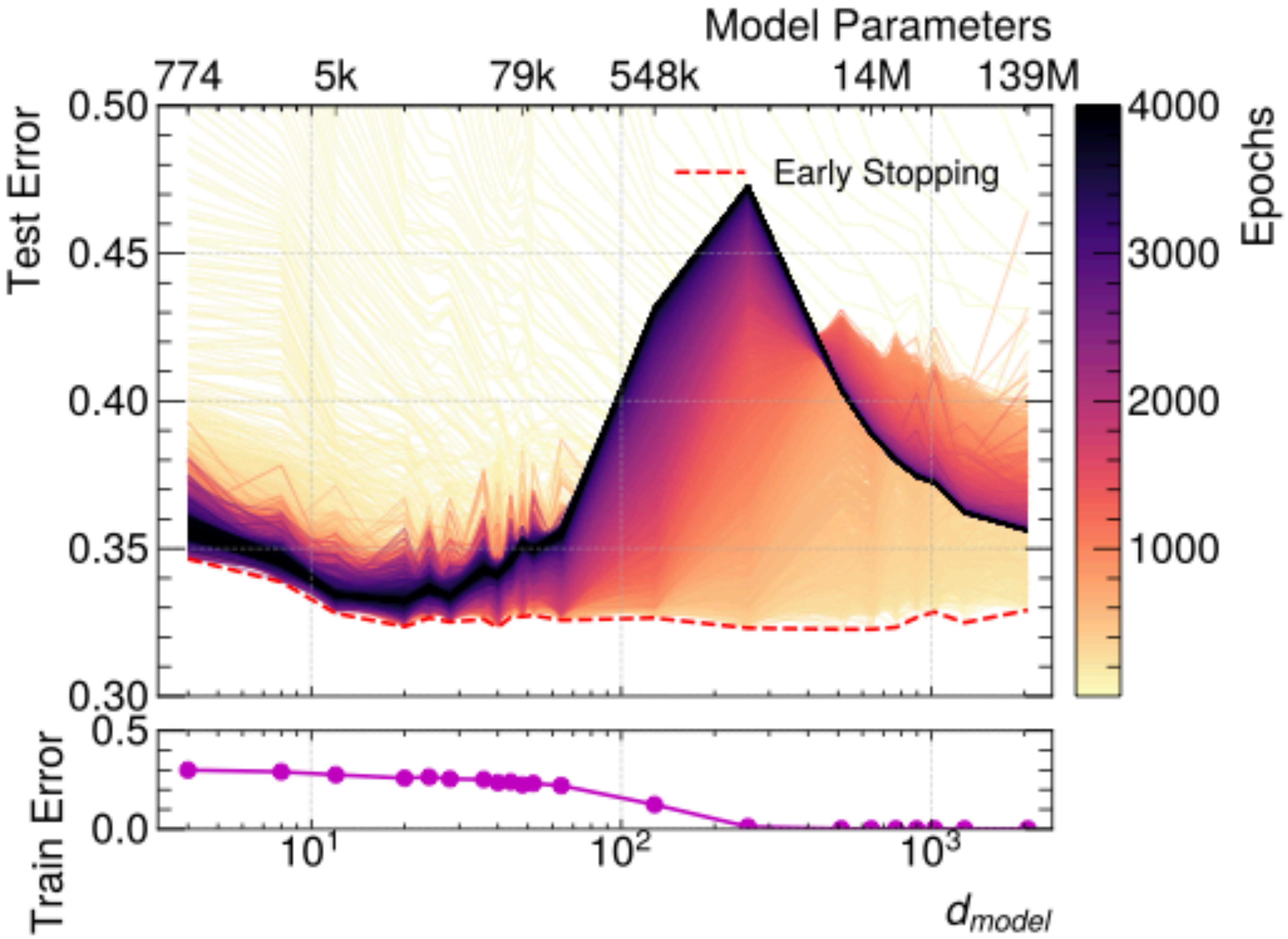
Probably less relevant for physics applications

- Likely our models not big enough
- Label noise not common

Double descent in HEP

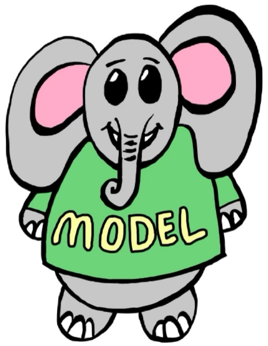
Training particle regression (jet pt)

- 200k jets (3 orders of magnitude smaller than SOTA datasets)
- order of mag more param than currently explored



Lesson: we are far from overparametrized in science!

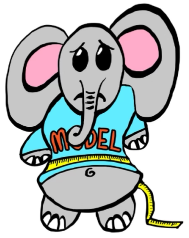
What did we learn today about choosing the right model?

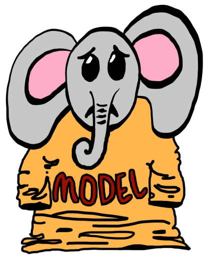


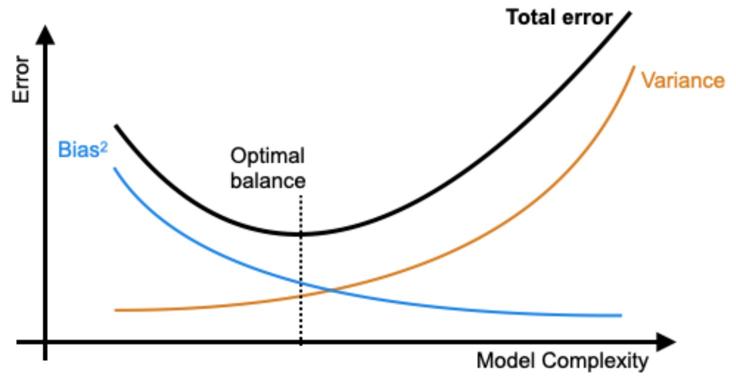
What did we learn today about choosing the right model?

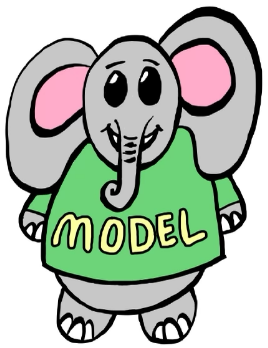
Total error = Bias² + Variance

Underfitting **Overfitting**





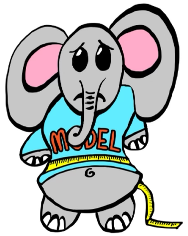


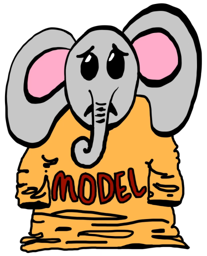


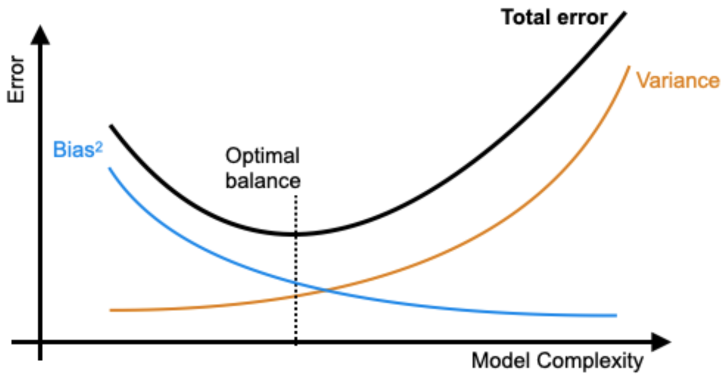
What did we learn today about choosing the right model?

Total error = Bias^2 + Variance

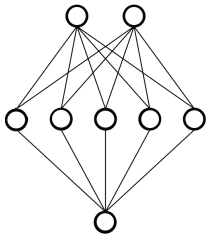
Underfitting **Overfitting**

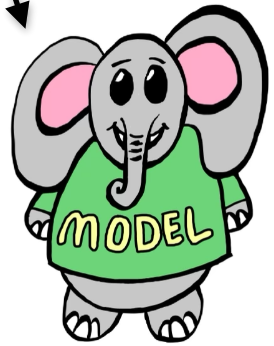






Deep learning gains:
increasing complexity

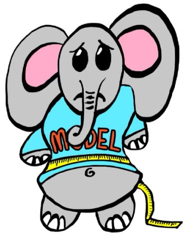


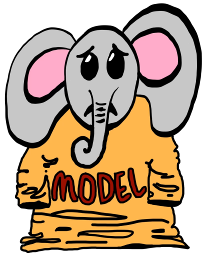


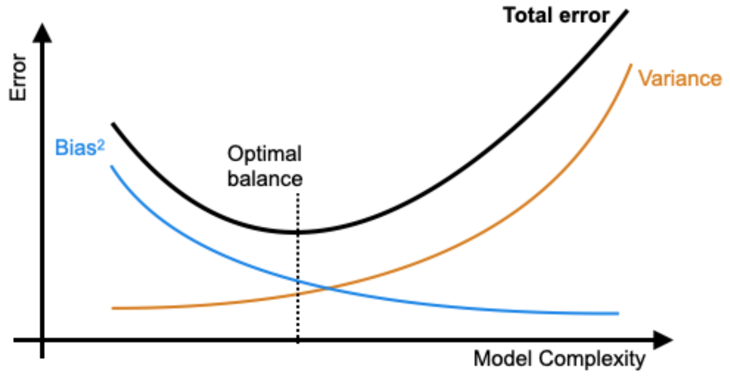
What did we learn today about choosing the right model?

Total error = Bias^2 + Variance

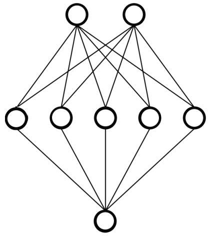
Underfitting **Overfitting**

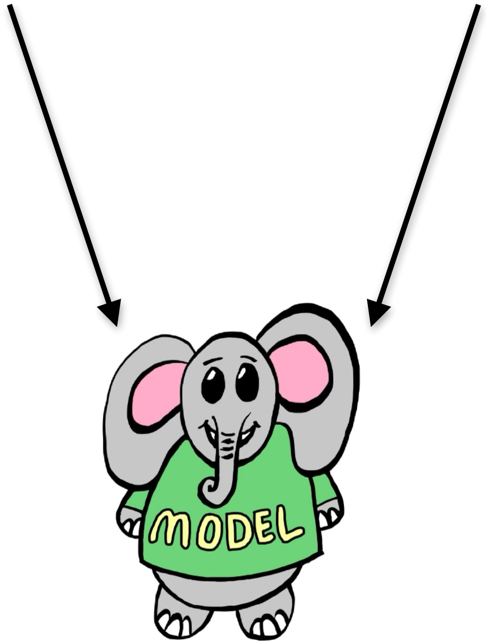






Deep learning gains:
increasing complexity





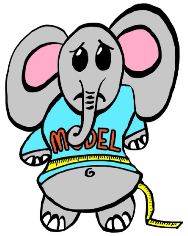
Regularization:

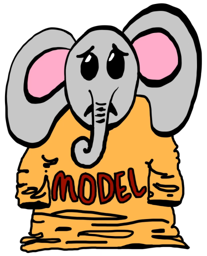
- ✓ L2 regularization
- ✓ Dropout
- ✓ Batch Norm
- ✓ Fine-tuning

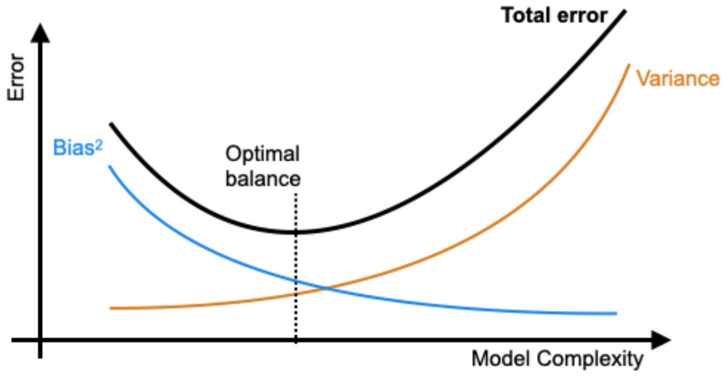
What did we learn today about choosing the right model?

Total error = Bias^2 + Variance

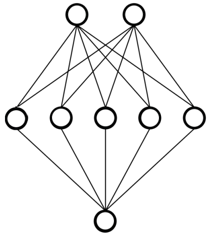
Underfitting **Overfitting**

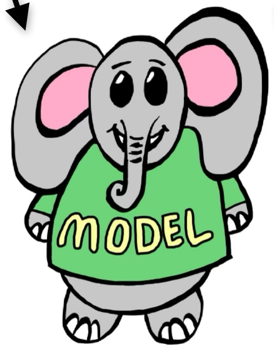






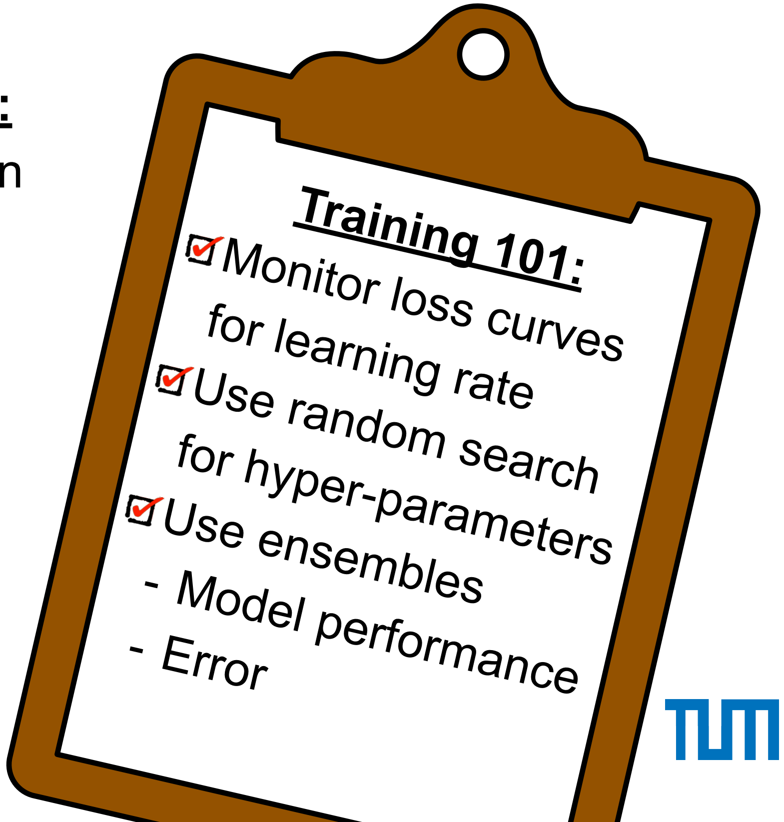
Deep learning gains: increasing complexity



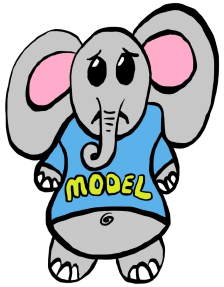


Regularization:

- ✓ L2 regularization
- ✓ Dropout
- ✓ Batch Norm
- ✓ Fine-tuning



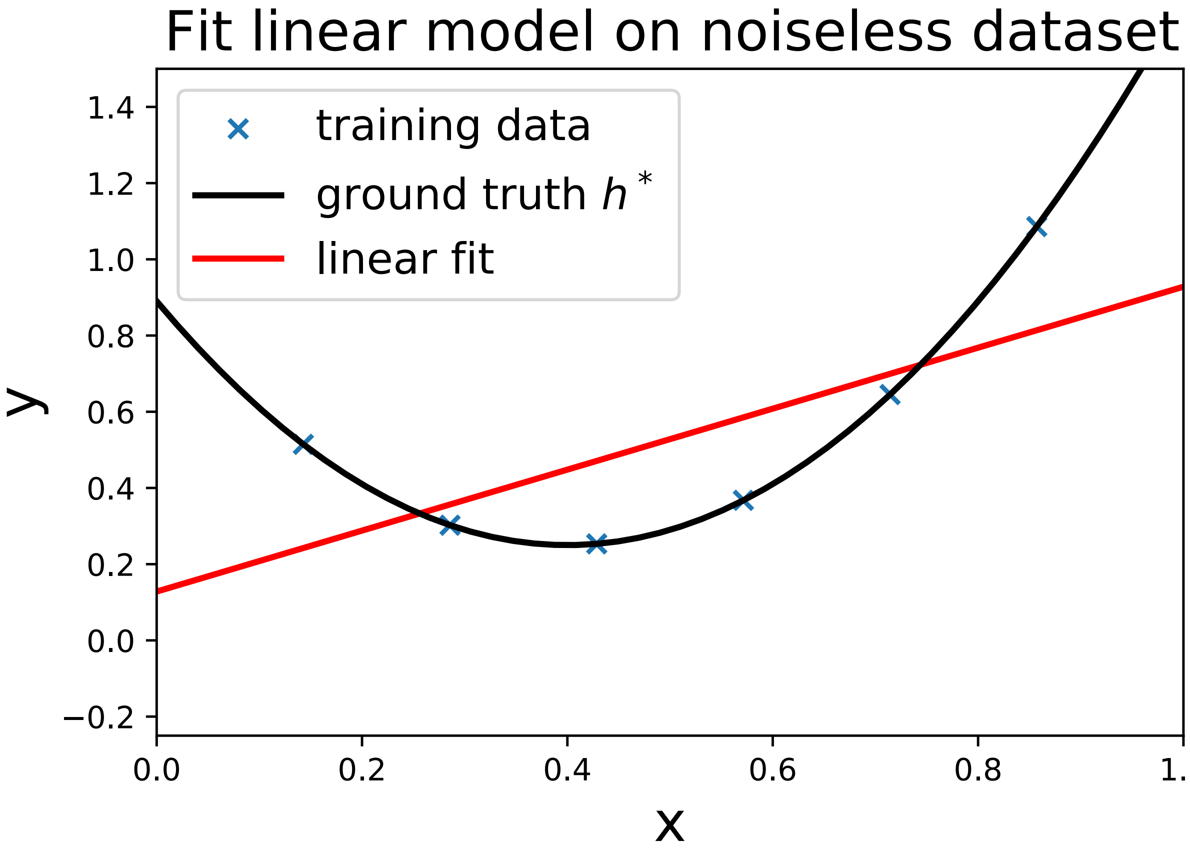
Backup



High bias: diagnostics

$$MSE(x) = \cancelto{0}{\sigma^2} + (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

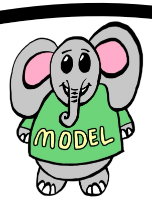
The training error on the linear model still large, even when there is *no noise* on the training data.



How to train?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

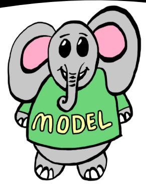


Train | Test

How to train?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$



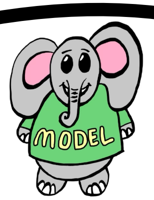
Train | Test

Issue: Don't want to look at the test set while optimizing!!

How to train?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$



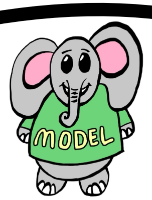
Issue: Don't want to look at the test set while optimizing!!

Train | Val | Test

How to train?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$



Issue: Don't want to look at the test set while optimizing!!

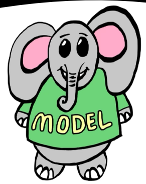
Train	Val	Test

Start with: 60 / 20 / 20

How to train?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$



Issue: Don't want to look at the test set while optimizing!!

Train	Val	Test

Start with: 60 / 20 / 20

Want a *large enough* validation set for statistically significant generalization metric.

Want a *large* training dataset to minimize $\mathscr{L}$.

Image by brgfx on Freepik

How to maximize statistics?

Minimize $\mathscr{L}$ by SGD

$$w = w - \alpha \nabla_w \mathscr{L}$$

How to maximize statistics?

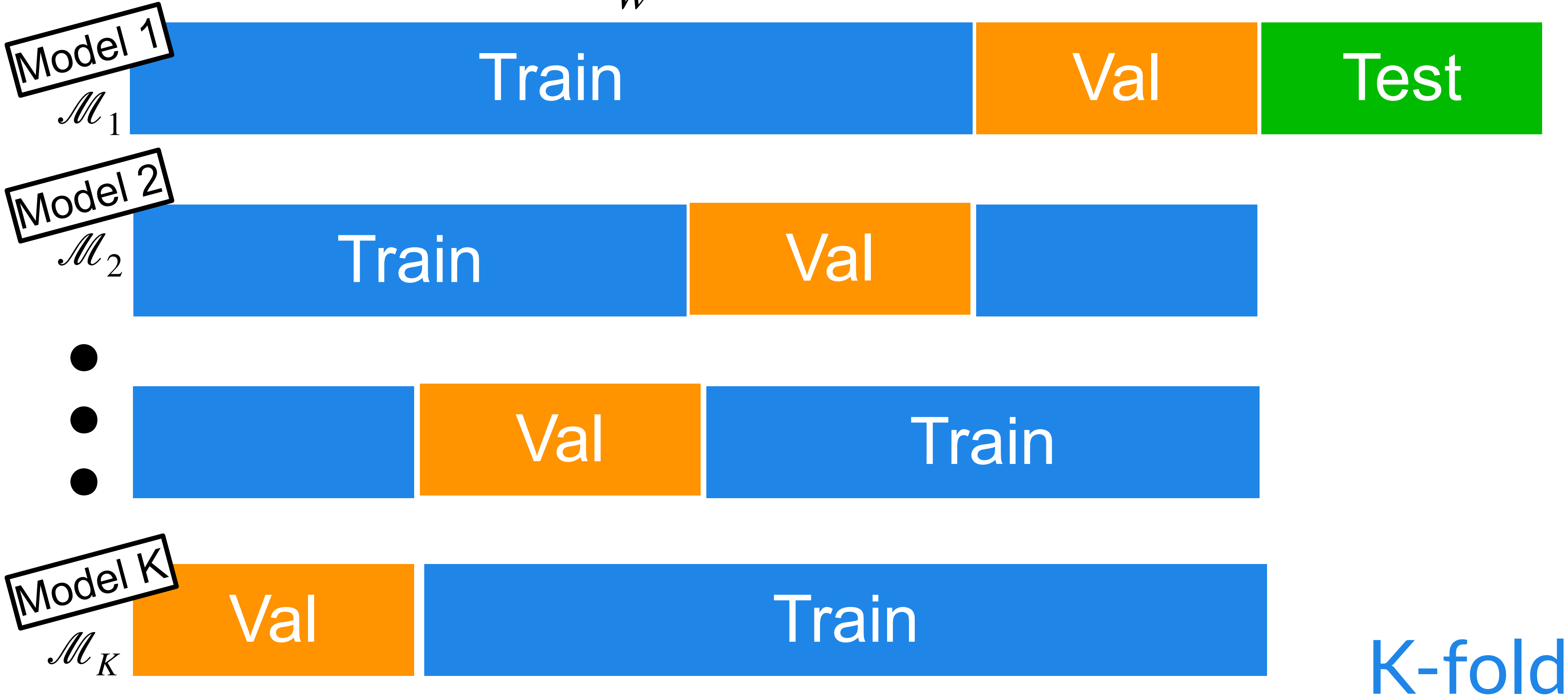
Minimize $\mathscr{L}$ by SGD on K splits for the dataset

$$w = w - \alpha \nabla_w \mathscr{L}$$

How to maximize statistics?

Minimize $\mathscr{L}$ by SGD on K splits for the dataset

$$w = w - \alpha \nabla_w \mathscr{L}$$

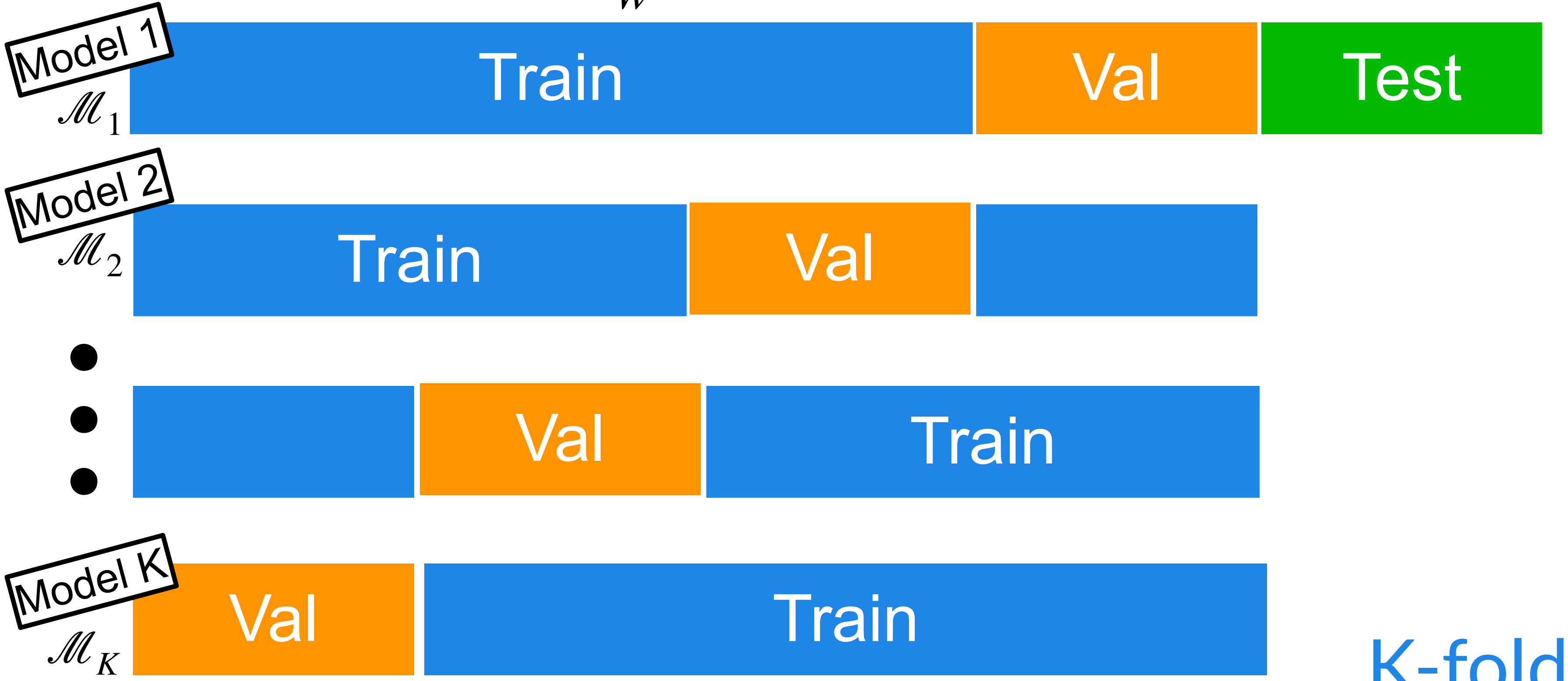


K-fold cross validation

How to maximize statistics?

Minimize $\mathcal{L}$ by SGD on K splits for the dataset

$$w = w - \alpha \nabla_w \mathcal{L}$$



At test time, *average* the predictions from the models

$$\mathcal{M} = \frac{1}{K} \sum_{i=1}^{K} \mathcal{M}_i$$

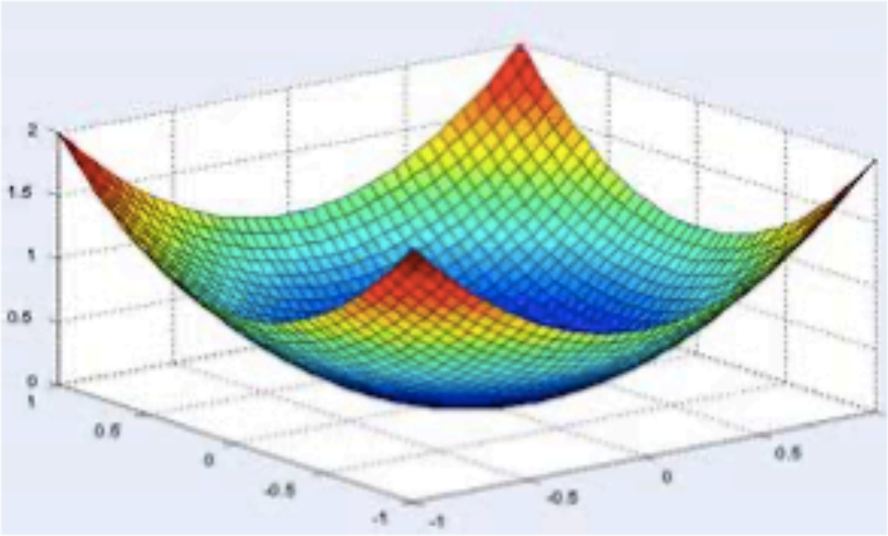
Ensembling helps performance!

- ✓ Couple percent gain in accuracy
- ✓ Used in Kaggle competitions!

K-fold cross validation

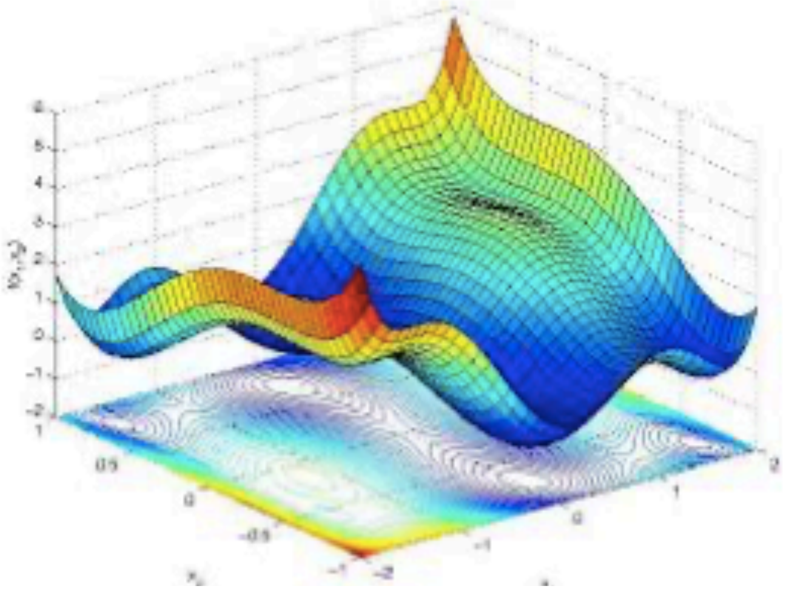
Loss landscape

Linear functions



(convex loss)

Neural networks



(non-convex)

Saliency maps

- NN: **nonlinear** function
- Approximate as a linear classifier by using a *Taylor expansion.*

$$S_c(I) \approx \theta^T I + b$$

$$\theta = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

Saliency maps

- NN: **nonlinear** function
- Approximate as a linear classifier by using a *Taylor expansion.*

$$S_c(I) \approx \theta^T I + b$$

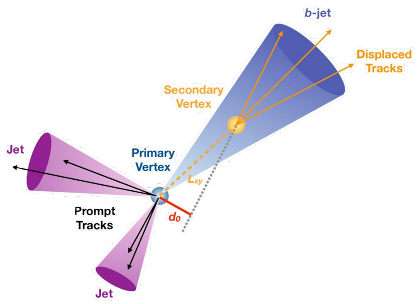
$$\theta = \left.\frac{\partial S_c}{\partial I}\right|_{I_0}$$

Saliency map: Plot the $|\theta|$ for each of these inputs

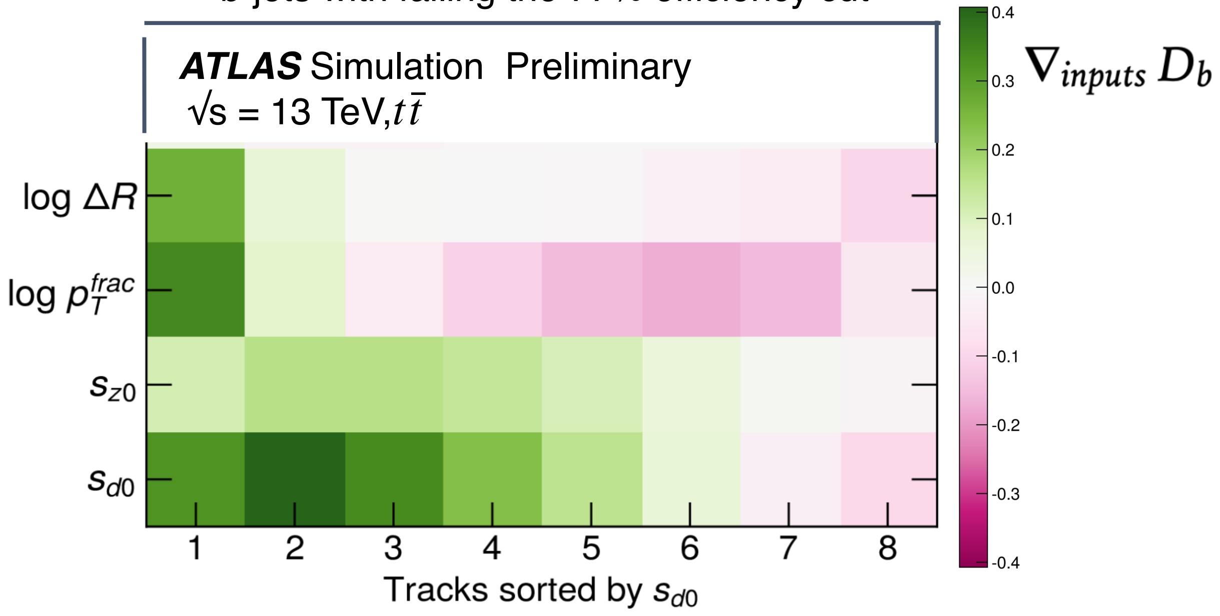
Saliency maps

Physics

Understand what the model has learned about this particle ID task.



b-jets with failing the 77% efficiency cut



ATL-PHYS-PUB-2020-014

Maths

Use saliency maps to postulate **new** conjectures which could then become new math theorems!

Article

Advancing mathematics by guiding human intuition with AI

https://doi.org/10.1038/s41586-021-04086-x
Received: 10 July 2021
Accepted: 30 September 2021
Published online: 1 December 2021
Open access
Check for updates

Alex Davies[1]✉, Petar Veličković[1], Lars Buesing[1], Sam Blackwell[1], Daniel Zheng[1], Nenad Tomašev[1], Richard Tanburn[1], Peter Battaglia[1], Charles Blundell[1], András Juhász[2], Marc Lackenby[2], Geordie Williamson[3], Demis Hassabis[1] & Pushmeet Kohli[1]✉

The practice of mathematics involves discovering patterns and using these to formulate and prove conjectures, resulting in theorems. Since the 1960s, mathematicians have used computers to assist in the discovery of patterns and formulation of conjectures[1], most famously in the Birch and Swinnerton-Dyer conjecture[2], a Millennium Prize Problem[3]. Here we provide examples of new fundamental results in pure mathematics that have been discovered with the assistance of machine learning—demonstrating a method by which machine learning can aid mathematicians in discovering new conjectures and theorems. We propose a process of using machine learning to discover potential patterns and relations between mathematical objects, understanding them with attribution techniques and using these observations to guide intuition and propose conjectures. We outline this machine-learning-guided framework and demonstrate its successful application to current research questions in distinct areas of pure mathematics, in each case showing how it led to meaningful mathematical contributions on important open problems: a new connection between the algebraic and geometric structure of knots, and a candidate algorithm predicted by the combinatorial invariance conjecture for symmetric groups[4]. Our work may serve as a model for collaboration between the fields of mathematics and artificial intelligence (AI) that can achieve surprising results by leveraging the respective strengths of mathematicians and machine learning.

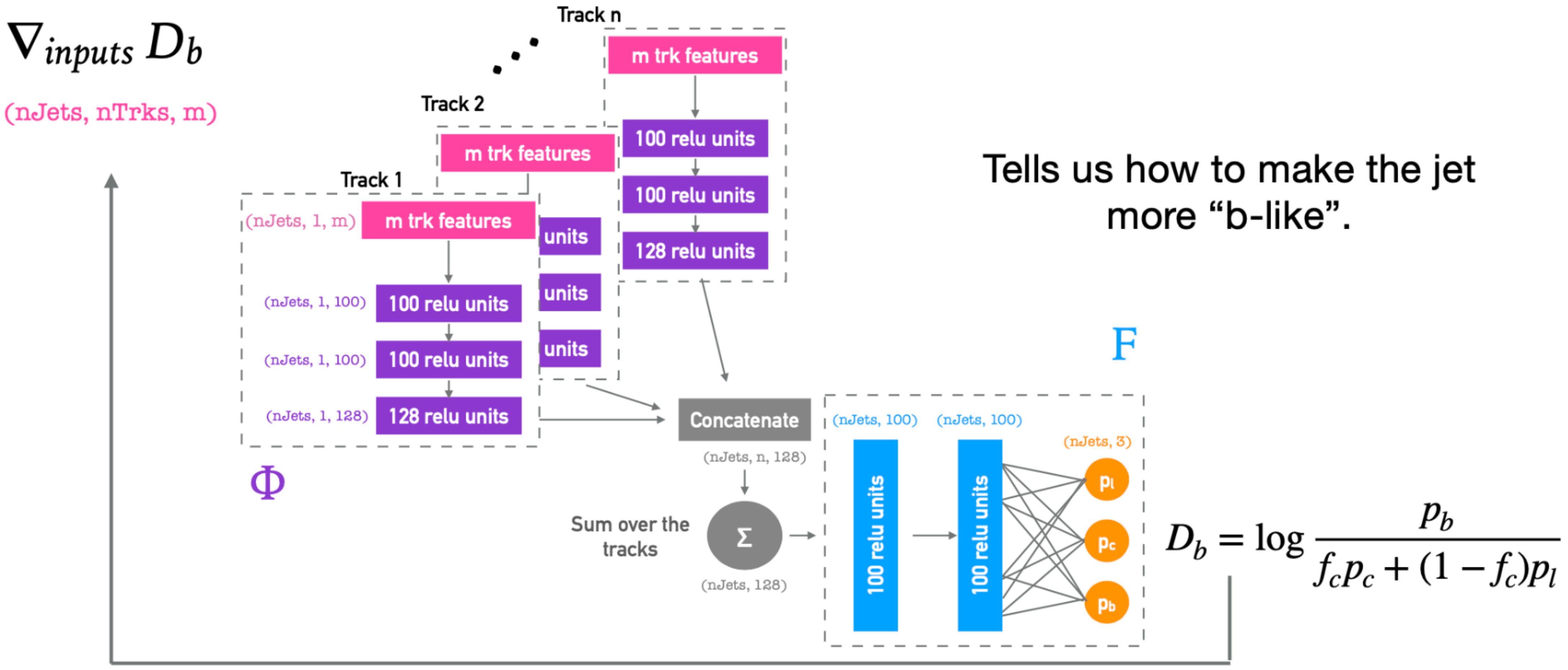
Nature 600, 70–74 (2021)

Saliency Maps

Model diagnostic

Saliency maps

What has DIPS learned about b-jets?



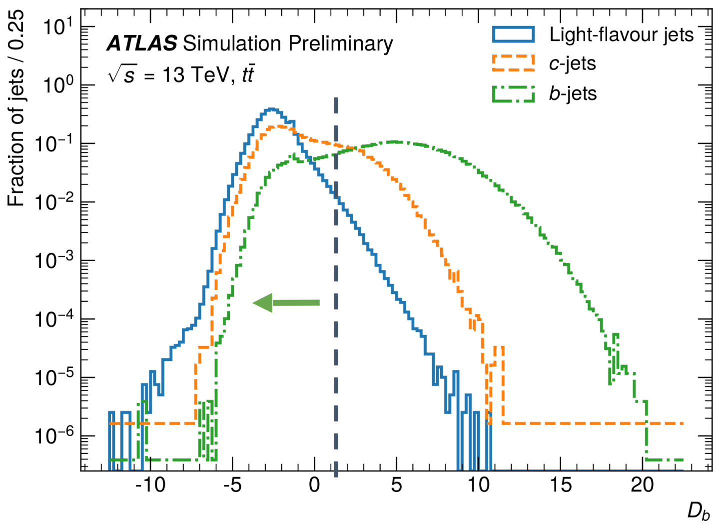
Tells us how to make the jet more “b-like”.

SLAC

Saliency definition

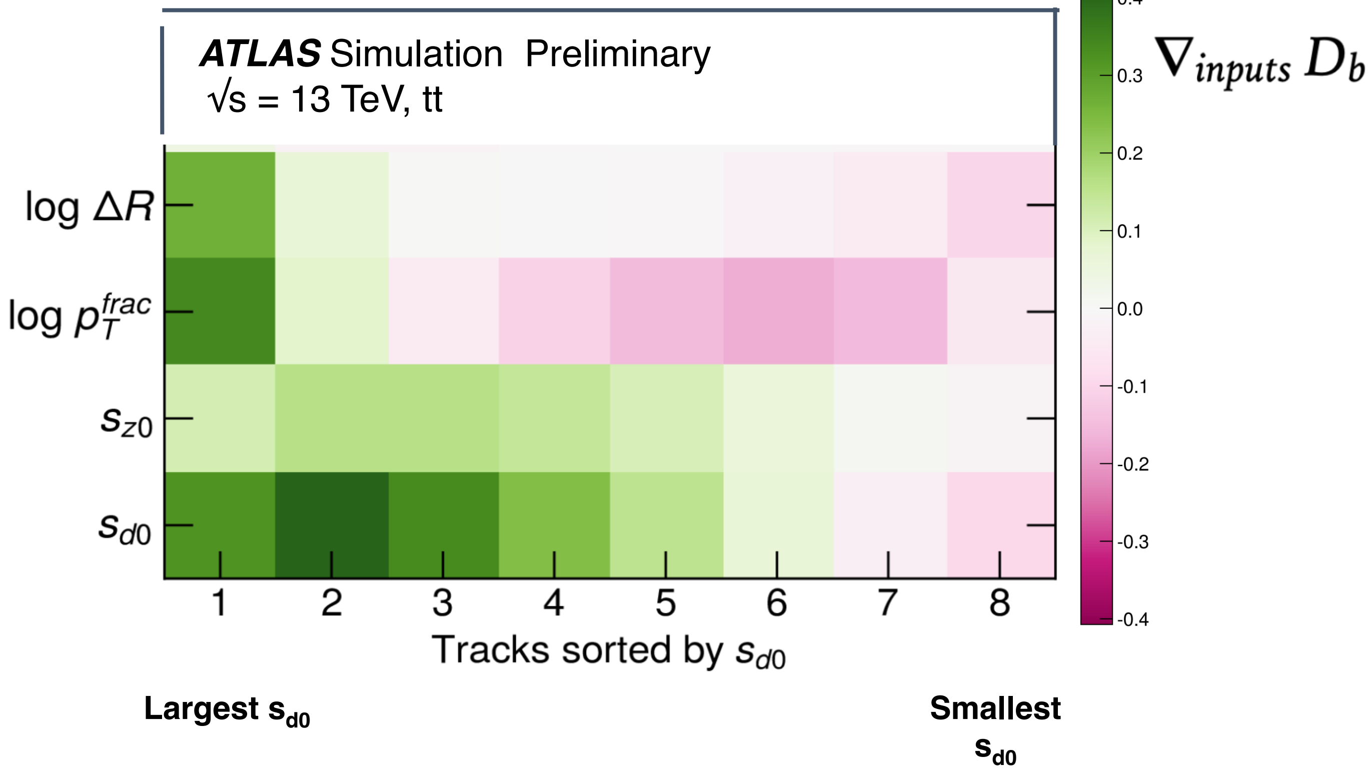
What has DIPS learned about b-jets?

- Consider b-jets failing the 77% WP



- Average over jets with 8 tracks
- Sort the tracks by s_{d0} for the average

b-jets with 8 tracks failing the 77% *b*-tagging working point

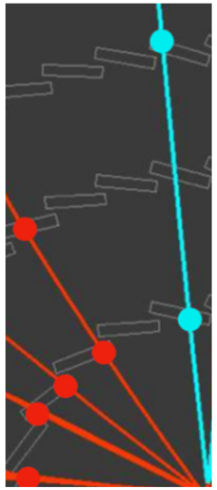


Saliency for DIPS

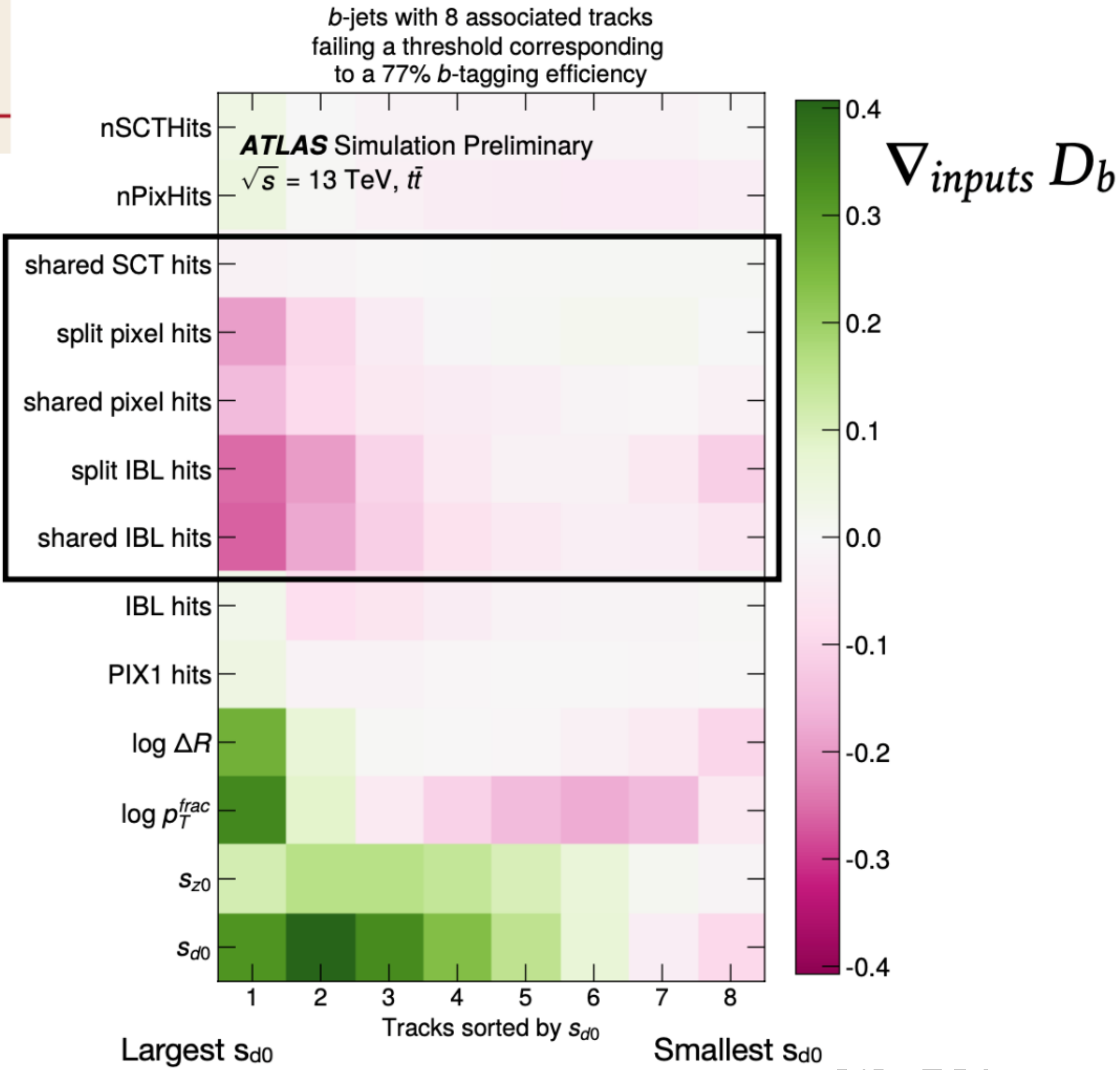
What has DIPS learned about b-jets?

- ✓ Want at least 5 high impact parameter tracks
- ✓ Wants harder leading track - as expected from b-decay
- ✓ Larger opening angle corresponding to geometrical constraint from more displaced tracks
- ✓ **Want good quality for displaced tracks**

Hit visualization



ATL-PHYS-PUB-2020-014



SLAC

2017: Stanford AI course

Motivation: virtual assistant

Tell information

Ask questions

Use natural language!

[demo]

Need to:

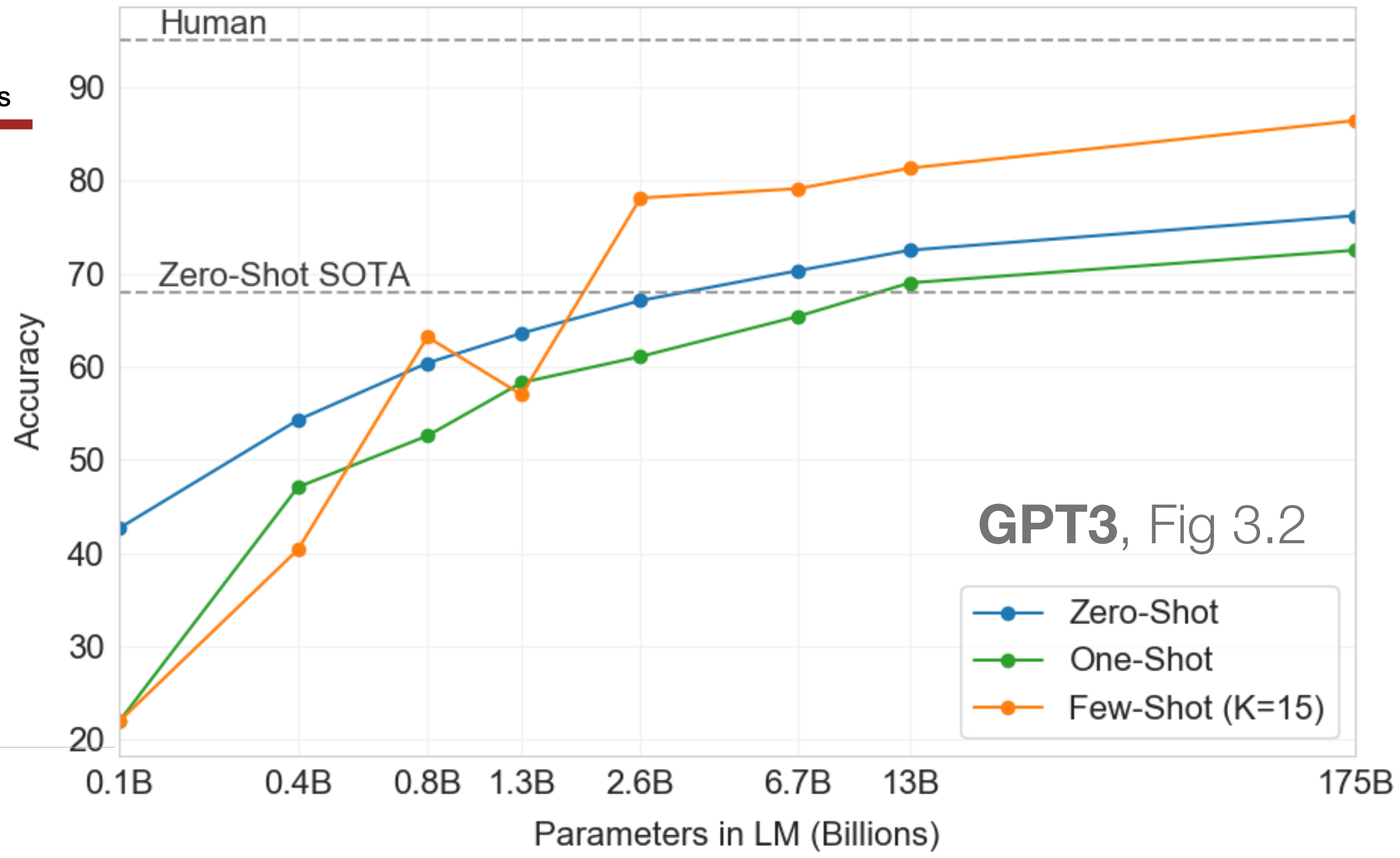
- Digest **heterogenous** information
- Reason **deeply** with that information

2020: Natural Language Processing

Tell information

Ask questions

Generative
Pre
Training



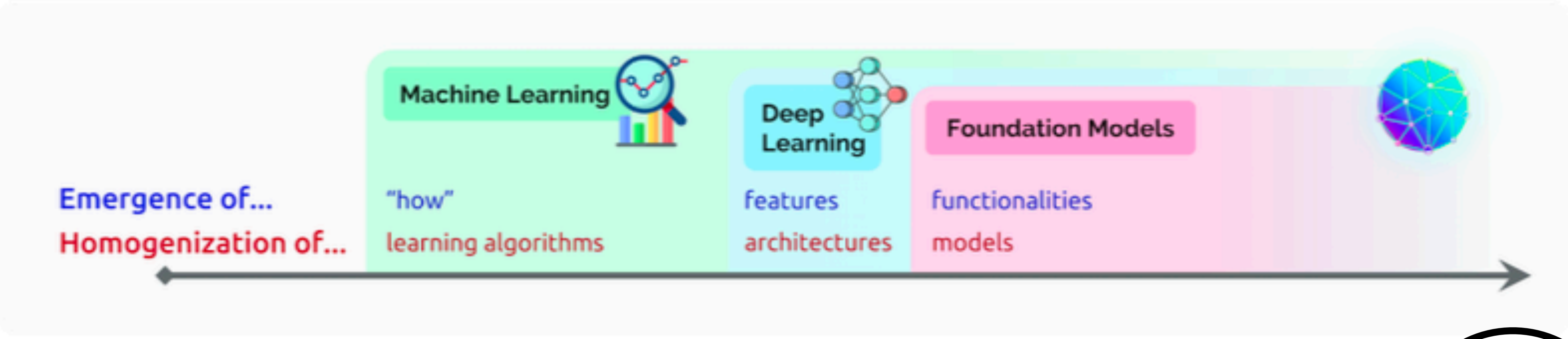
GPT3, Fig 3.2

Comparing w/ size of
SOTA jet tagging models

GN2 (2m)
2023

GN3 (13m)
2025

2021: Foundation models



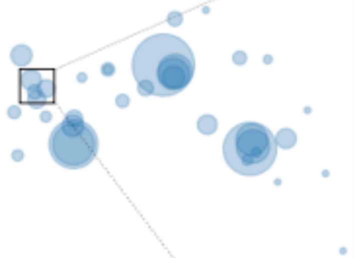
“We introduce the term **foundation models** to fill a void in describing the p
shift we are witnessing…”

Percy Liang contact author

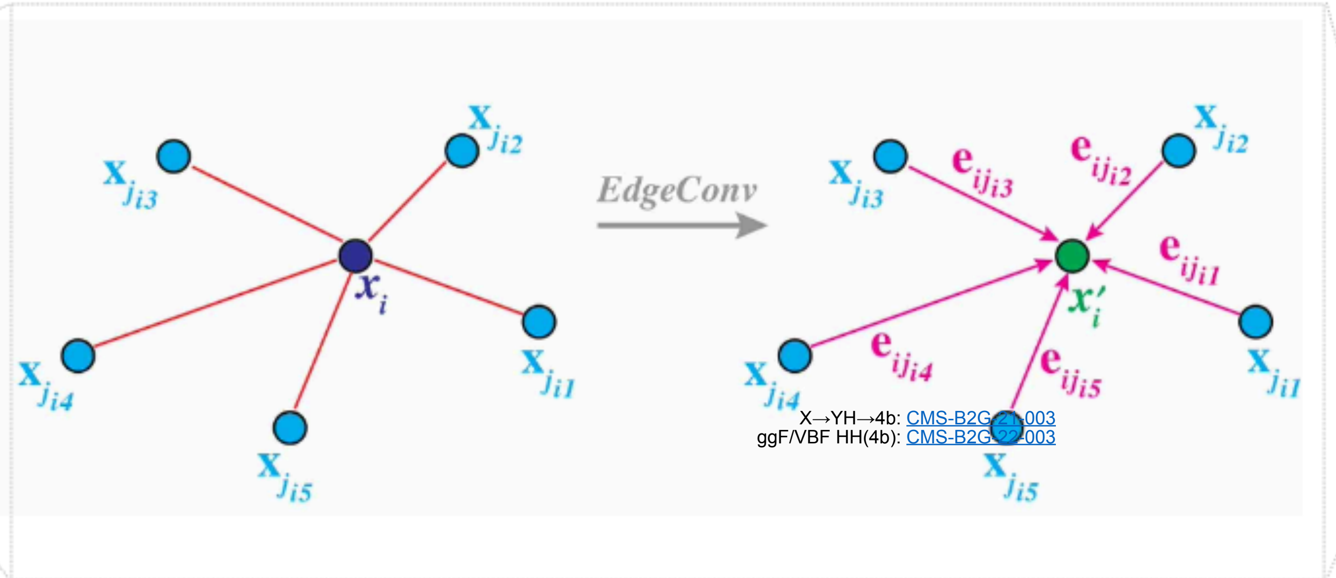
“From a technological point of view, foundation models are not new… however, the sheer scale and scope o
foundation models from the last few years have stretched our imagination of what is possible.”

Dynamic Graph CNN

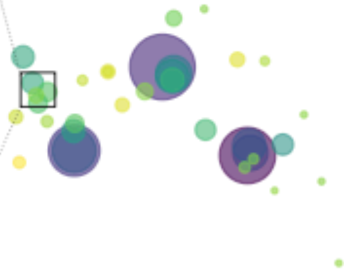
“ParticleNet” [CMS]



Model the **jet** as a point cloud



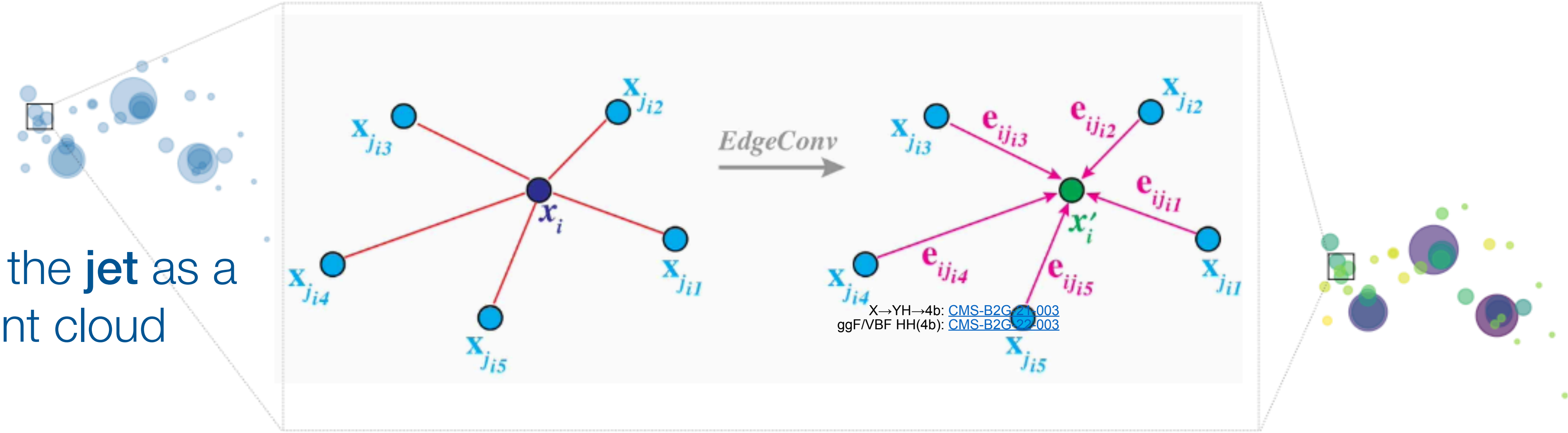
X→YH→4b: CMS-B2G-21-003
ggF/VBF HH(4b): CMS-B2G-22-003



Dynamic Graph CNN

"ParticleNet" [CMS]

Model the **jet** as a point cloud

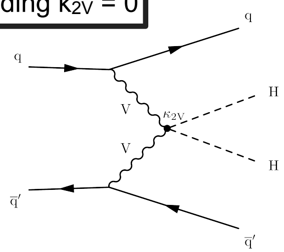


X→YH→4b: CMS-B2G-21-003
ggF/VBF HH(4b): CMS-B2G-22-003

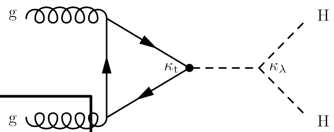
Very impressive physics results
But was the graph representation needed?

VBF HH→4b

First analysis excluding $\kappa_{2V} = 0$



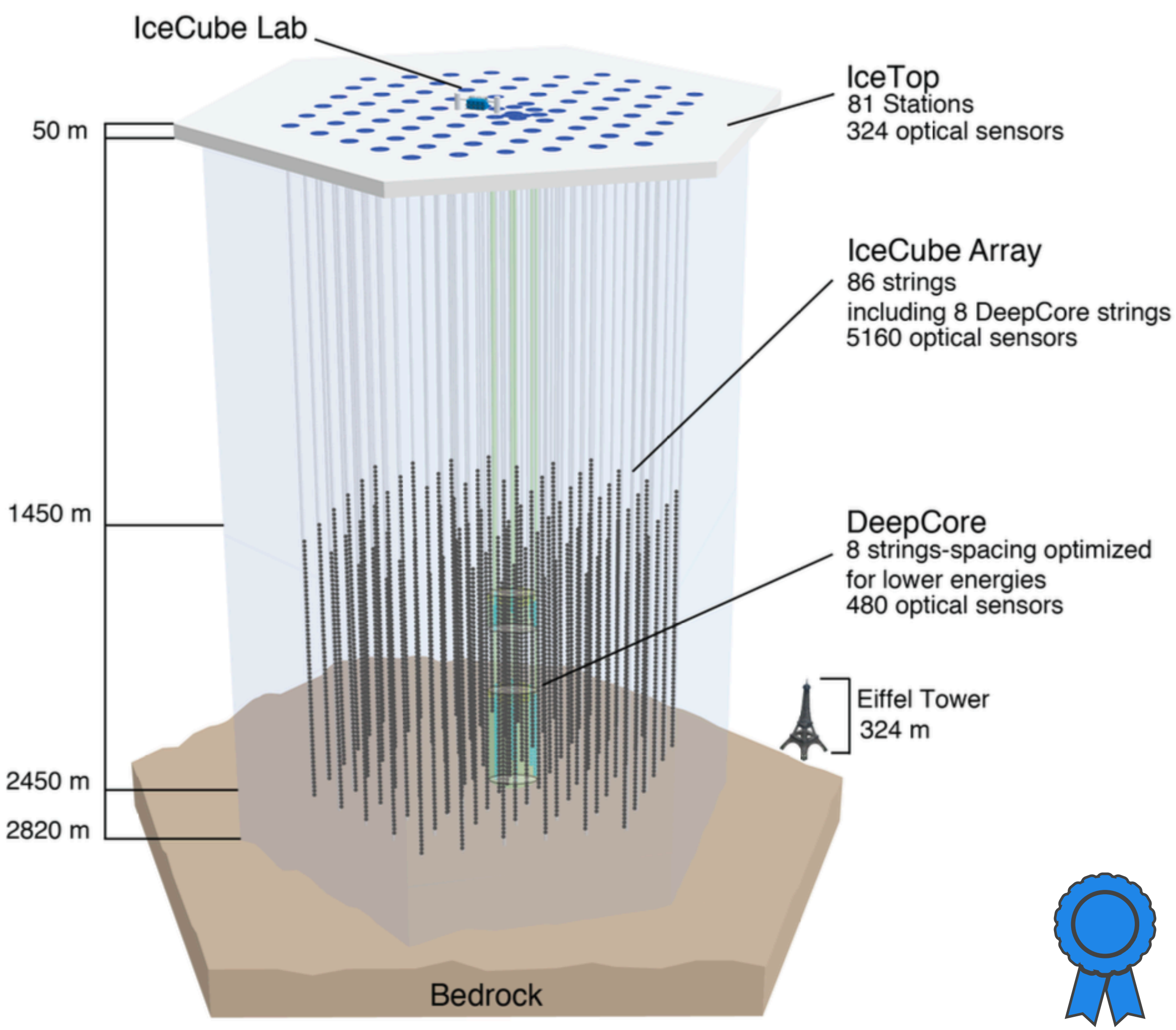
ggF HH→4b



Boosted:
Obs (exp): 9.9 (5.1)

Resolved:
ATLAS Obs (exp): 5.4 (8.1)
CMS Obs (exp): 3.8 (7.8)


Quarks
u c t
up charm top
d s b
down strange bottom
Forces
Z γ
Z boson photon
W g
W boson gluon
H
Higgs boson
e μ τ
electron muon tau
νe νμ ντ
electron neutrino muon neutrino tau neutrino
Leptons
How did this story evolve for neutrino identification?



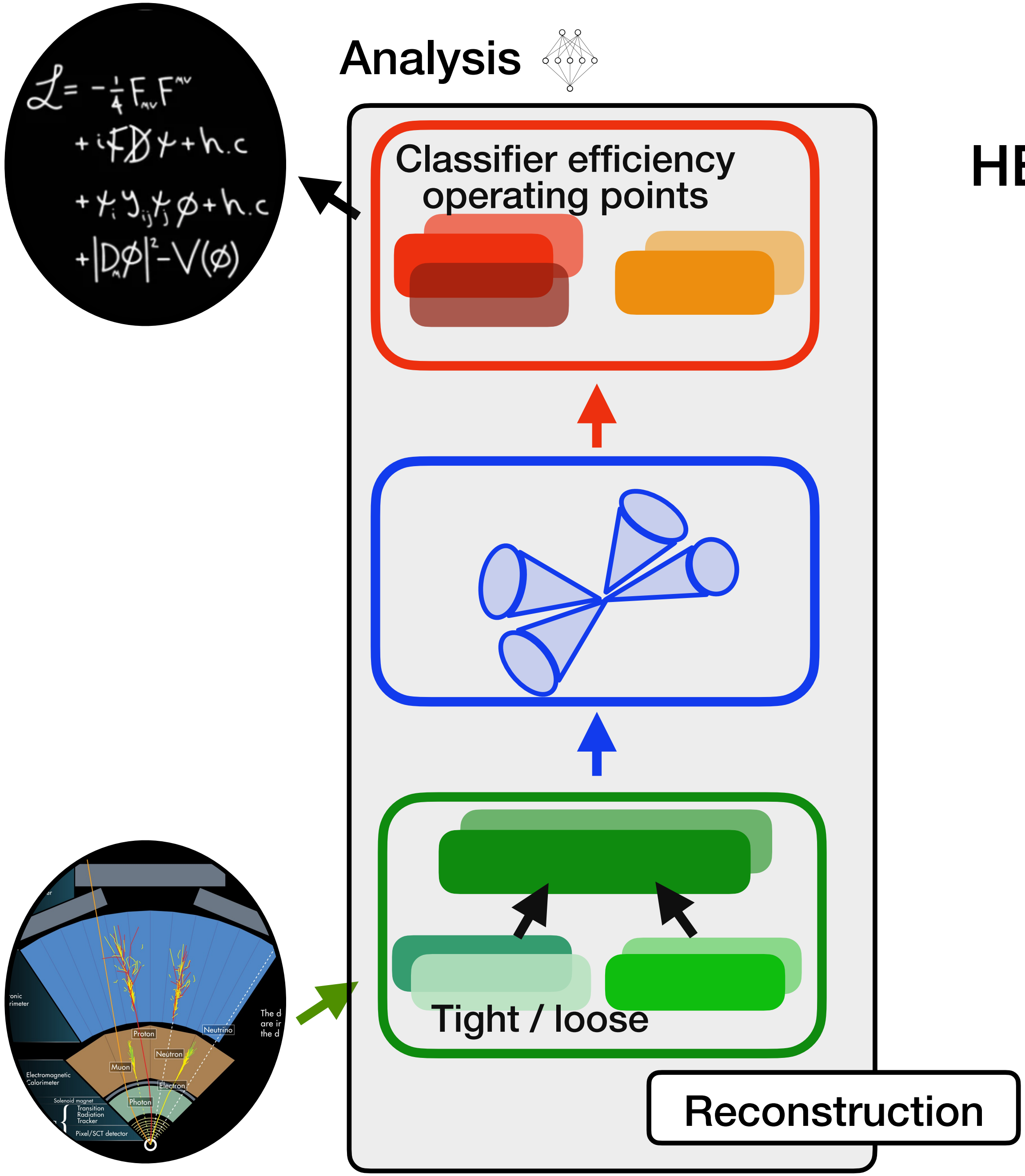
kaggle

3.1.4 Edge Selection

In the original EdgeConv implementation, edges are calculated in each layer dynamically by *k*-Nearest Neighbors (*k*NN). However, this edge selection scheme is not differentiable in itself and therefore does not have gradients. This would still work well for the segmentation task in the original paper, as points in the same segment are trained to be close in the latent space. However, the situation is different in this task, where it would not make sense to dynamically select edges. Therefore, the edges used in EdgeConv are calculated from the input features only once in our model.

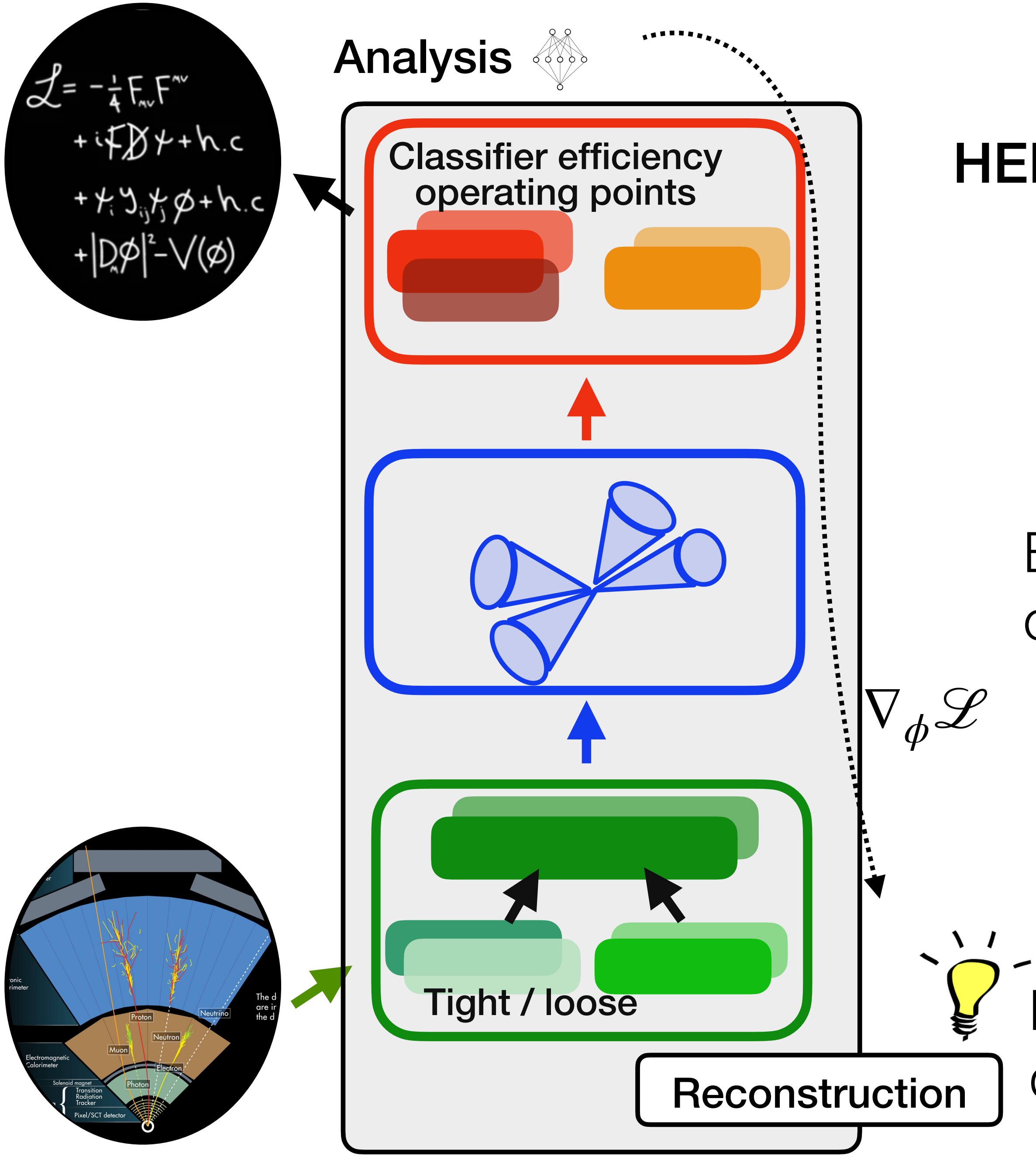
All three of the winning solutions used a transformer architecture.

Also in jet tagging (ATLAS and CMS), transformers outperform GNN architectues.



HEP reconstruction is trained on broad data

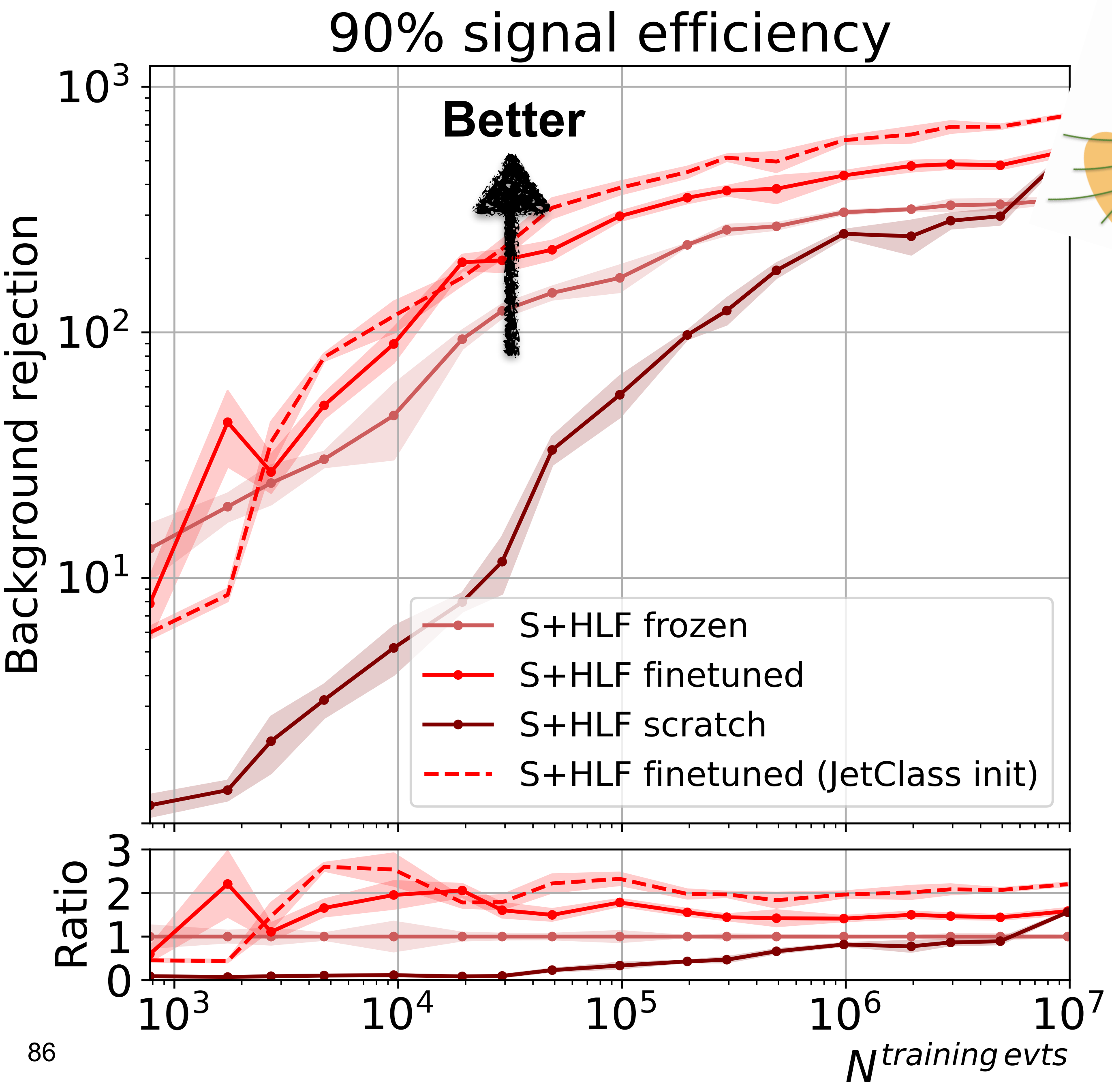
Each of analysis chooses their own operating points (grad student descent).

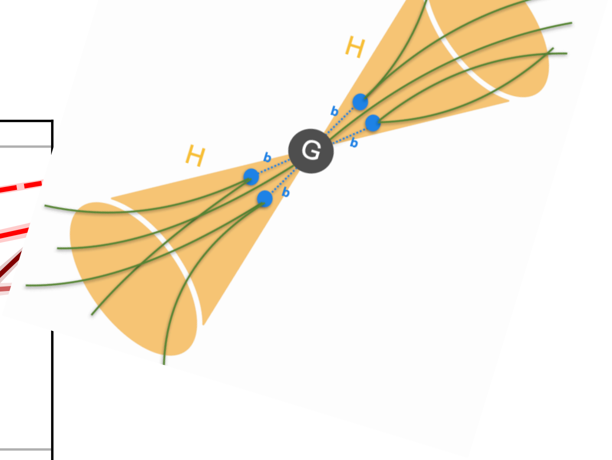


HEP reconstruction is trained on broad data

Each of analysis chooses their own operating points (grad student descent).

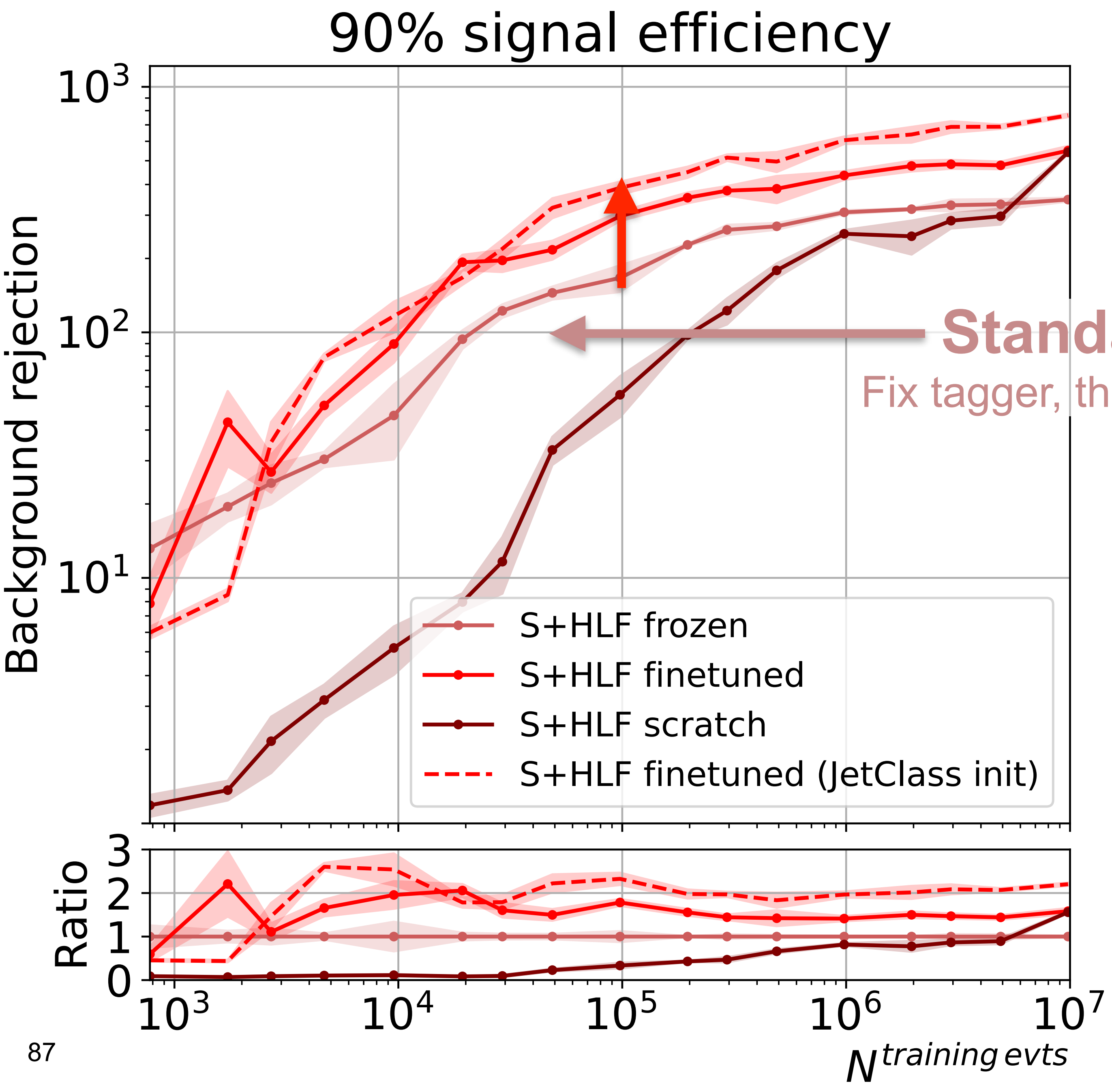
Foundation models: can we let gradients do our work for us? **“Fine-tuning”**





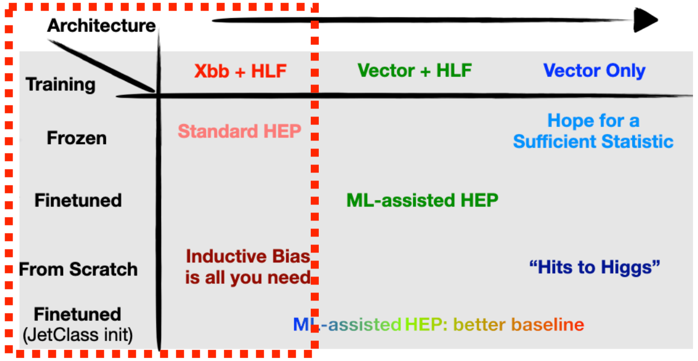
How to read?

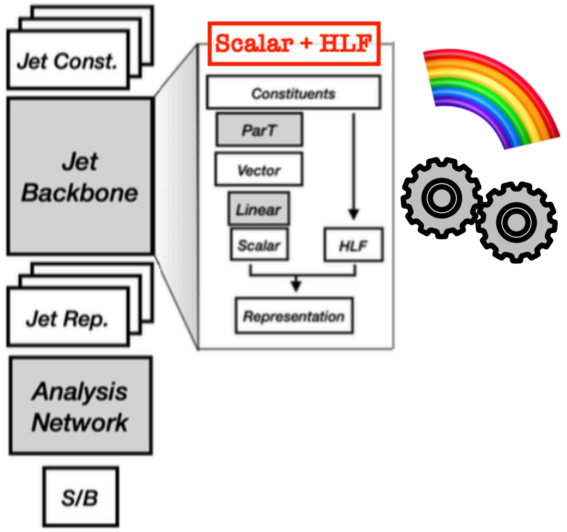
$$\text{background rejection} = \frac{1}{\text{background}}$$



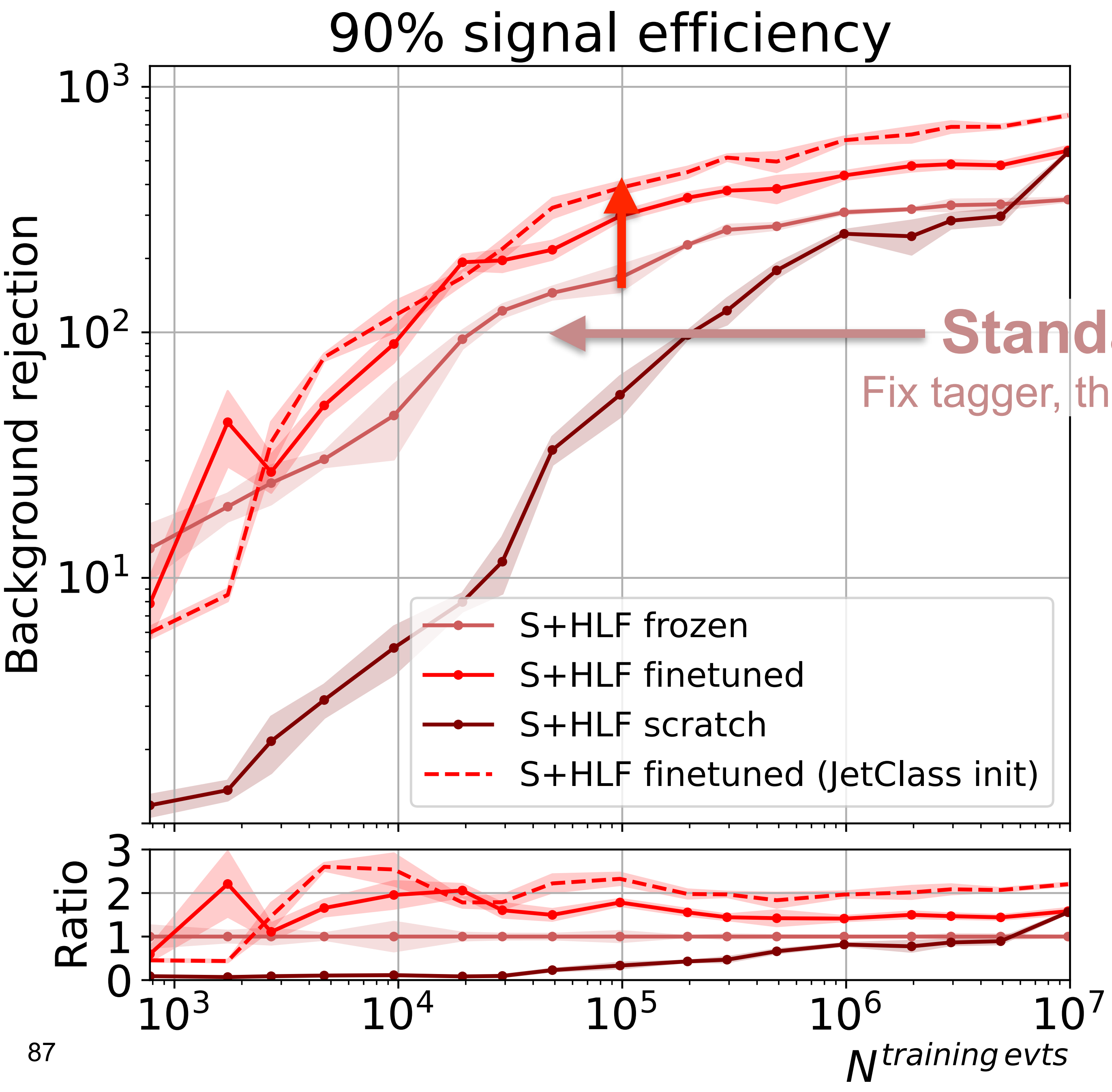
Standard HEP

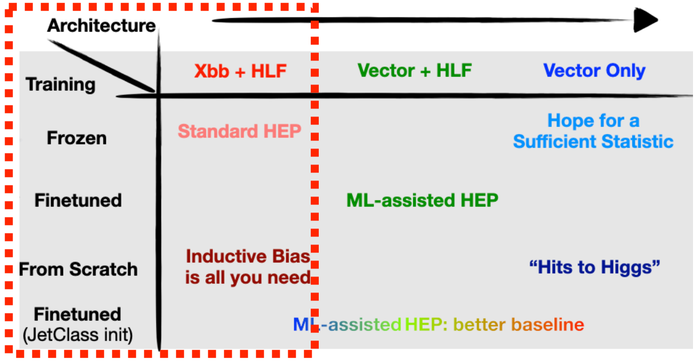
Fix tagger, then train analysis





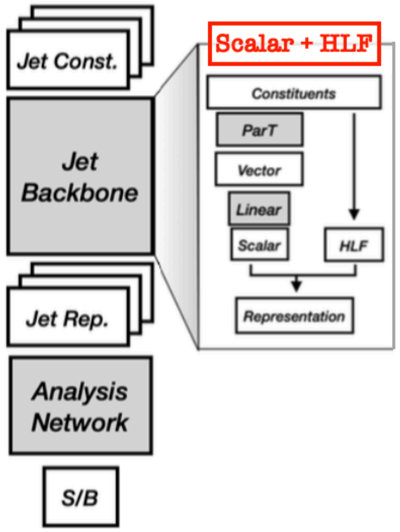
A better baseline tagger improves the analysis even more (!)





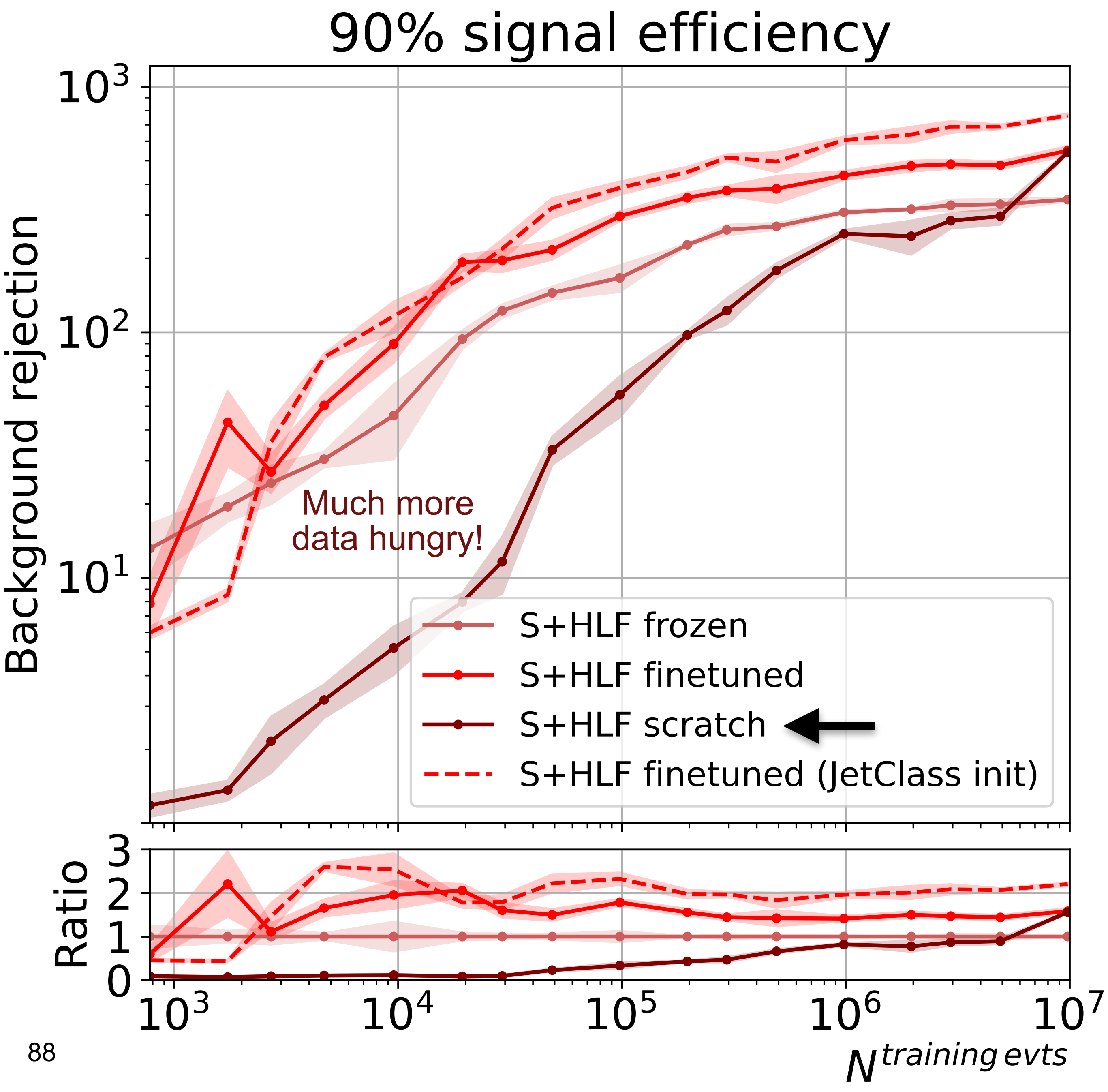
Standard HEP

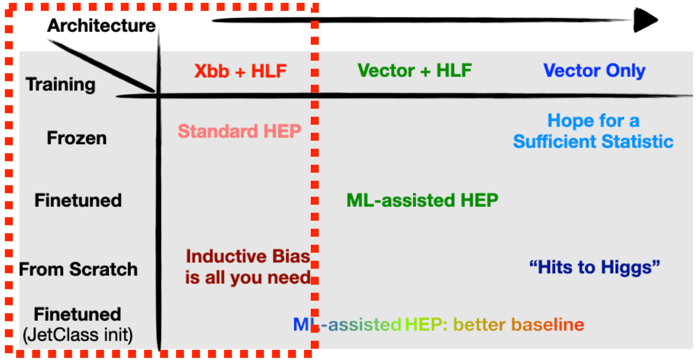
Fix tagger, then train analysis

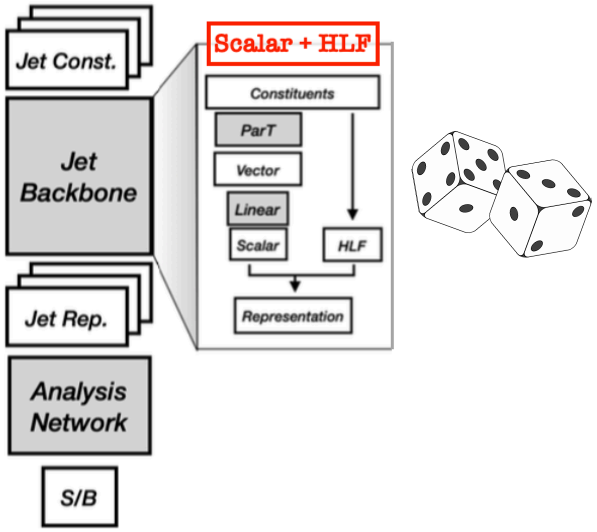


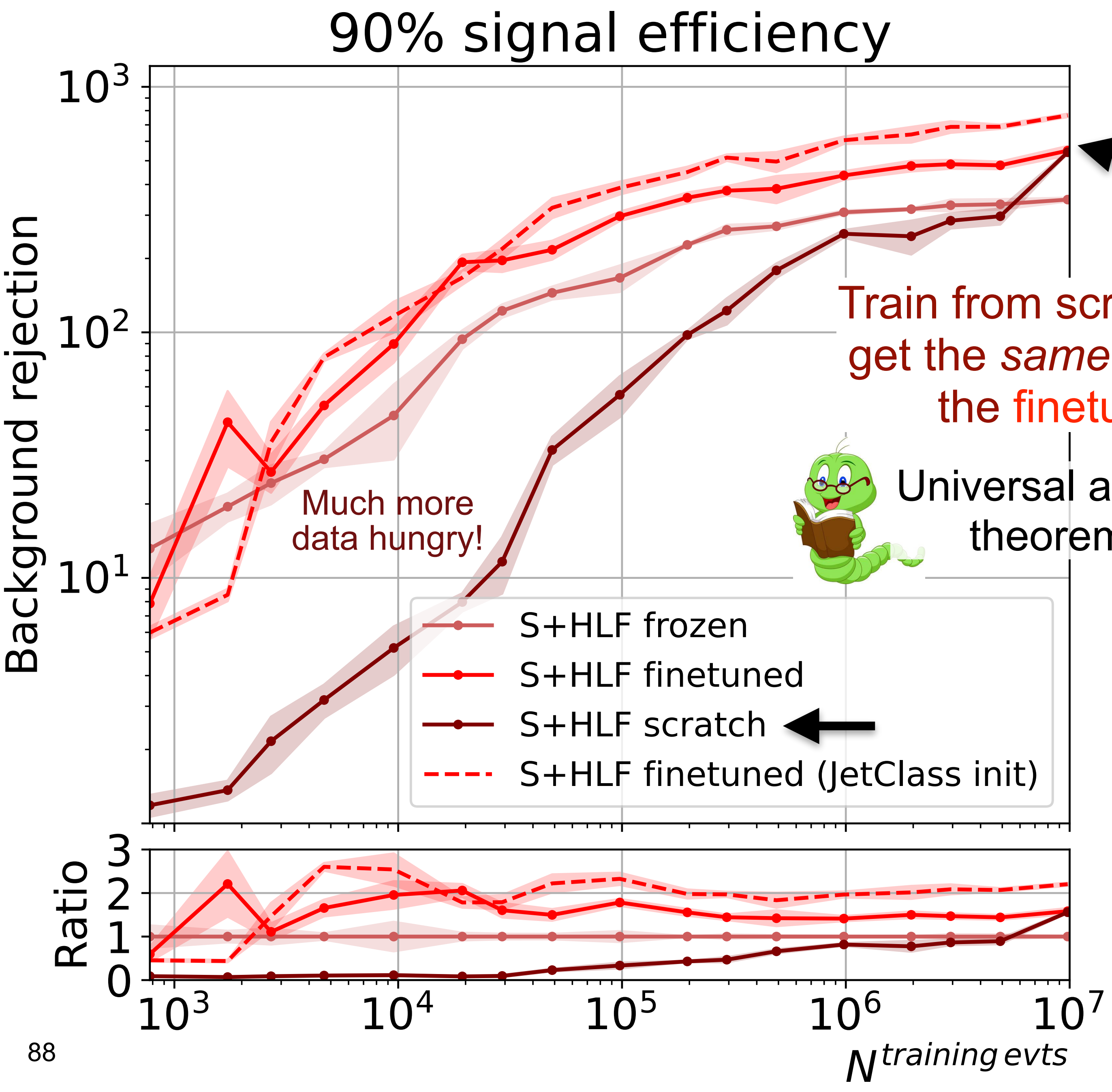
A better baseline tagger improves the analysis even more (!)

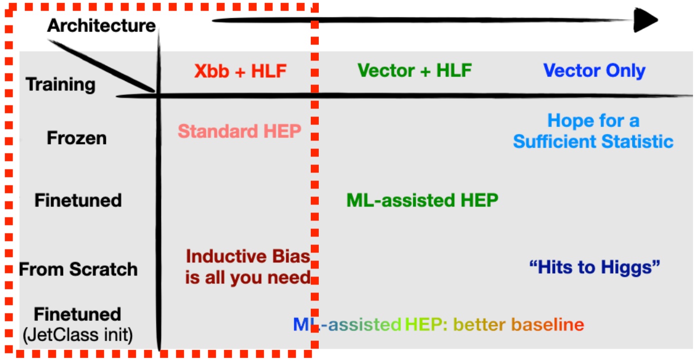
How do build the foundation model


90% signal efficiency
Background rejection
Ratio
Much more data hungry!
S+HLF frozen
S+HLF finetuned
S+HLF scratch
S+HLF finetuned (JetClass init)
$N^{training\,evts}$


Architecture
Training
Xbb + HLF
Vector + HLF
Vector Only
Frozen
Standard HEP
Hope for a Sufficient Statistic
Finetuned
ML-assisted HEP
From Scratch
Inductive Bias is all you need
"Hits to Higgs"
Finetuned (JetClass init)
ML-assisted HEP: better baseline

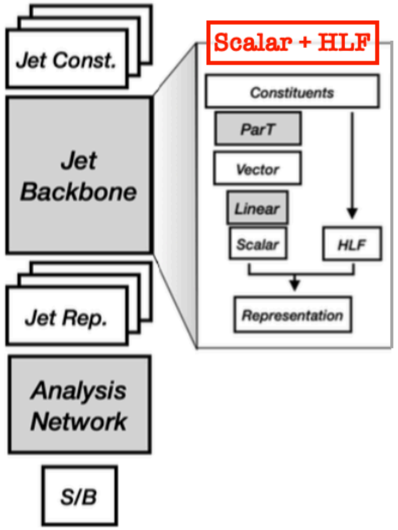

Jet Const.
Jet Backbone
Jet Rep.
Analysis Network
S/B
Scalar + HLF
Constituents
ParT
Vector
Linear
Scalar
HLF
Representation

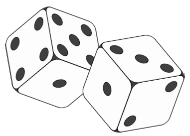


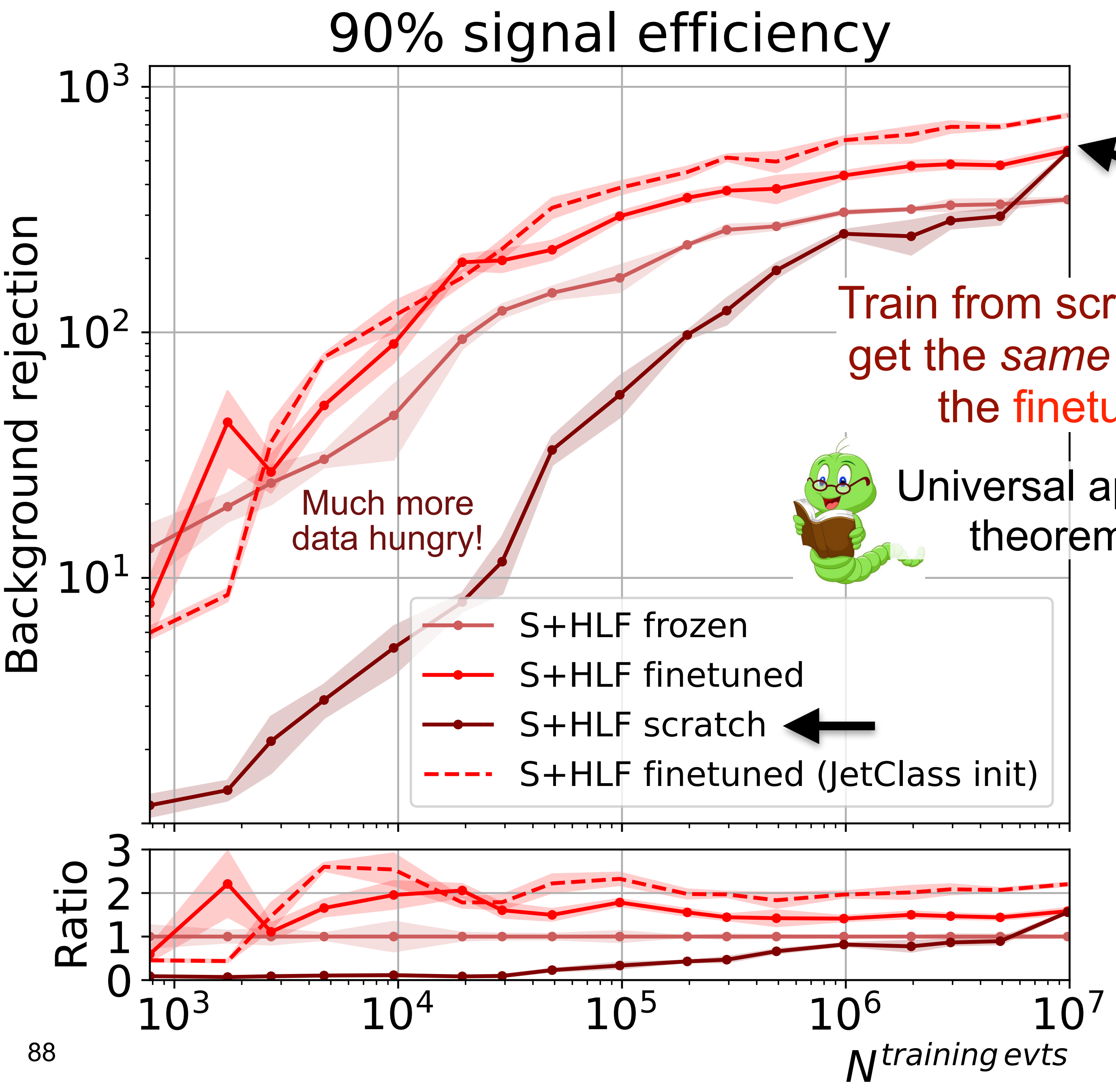


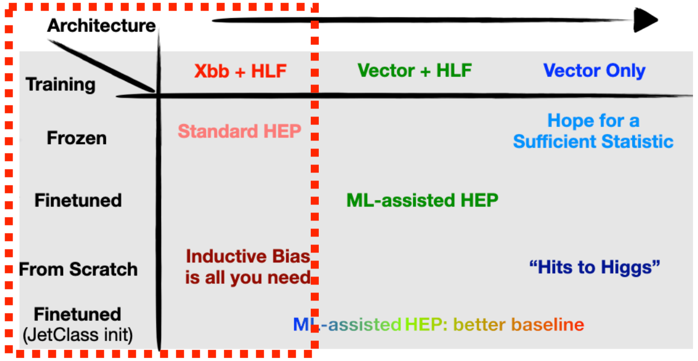
Train from scratch (eventually) get the *same performance* as the finetuned model.

Universal approximation theorem for NNs



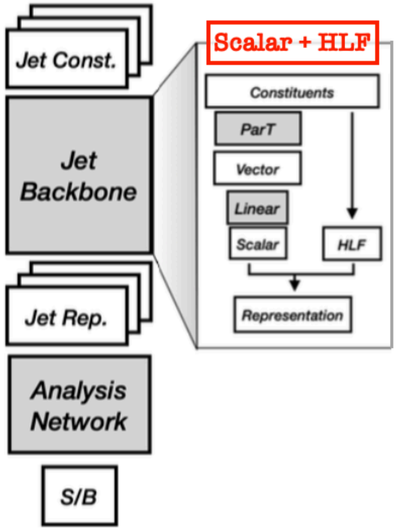


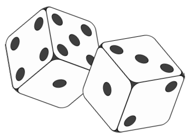




Train from scratch (eventually) get the *same performance* as the finetuned model.

Universal approximation theorem for NNs

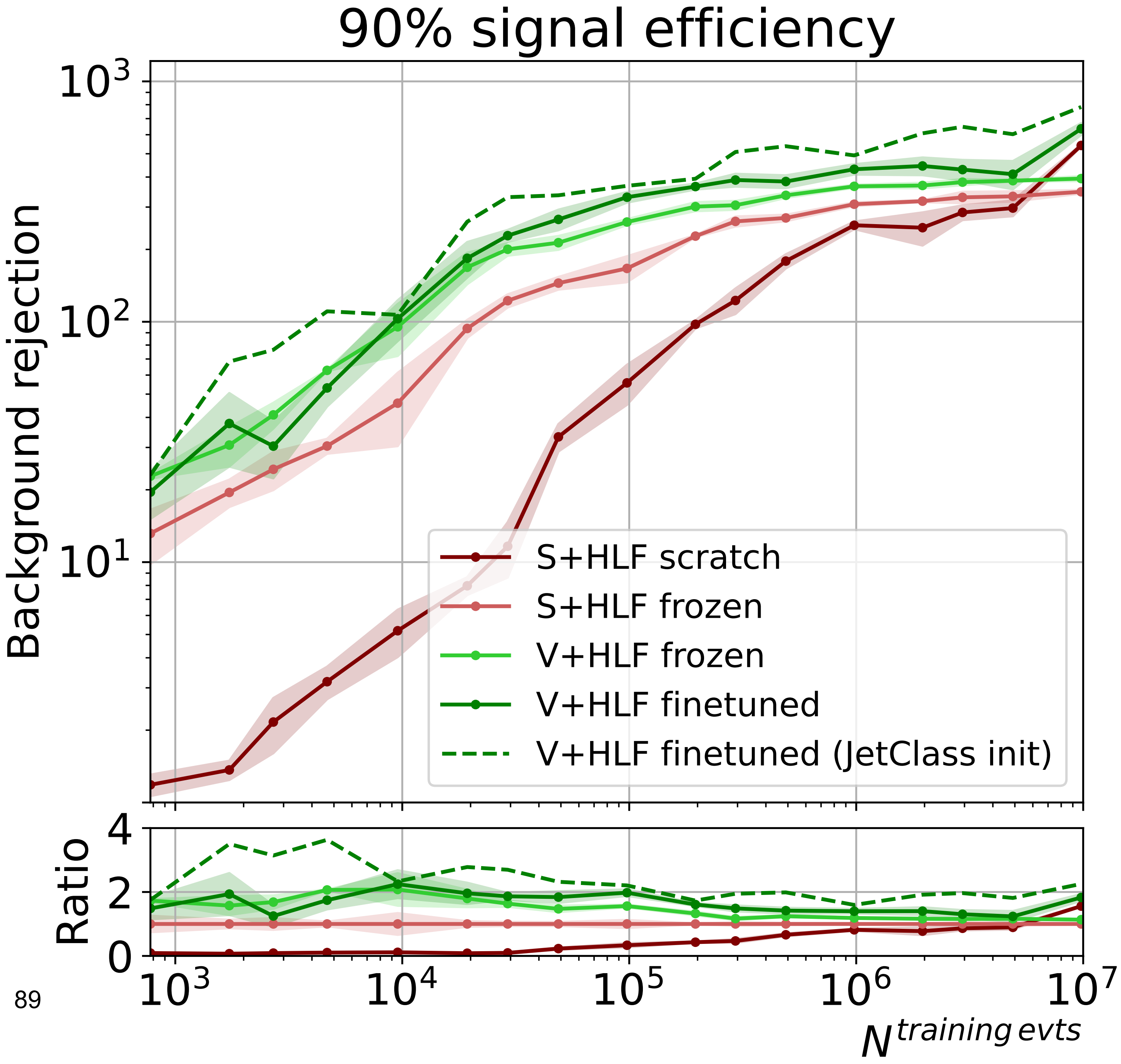




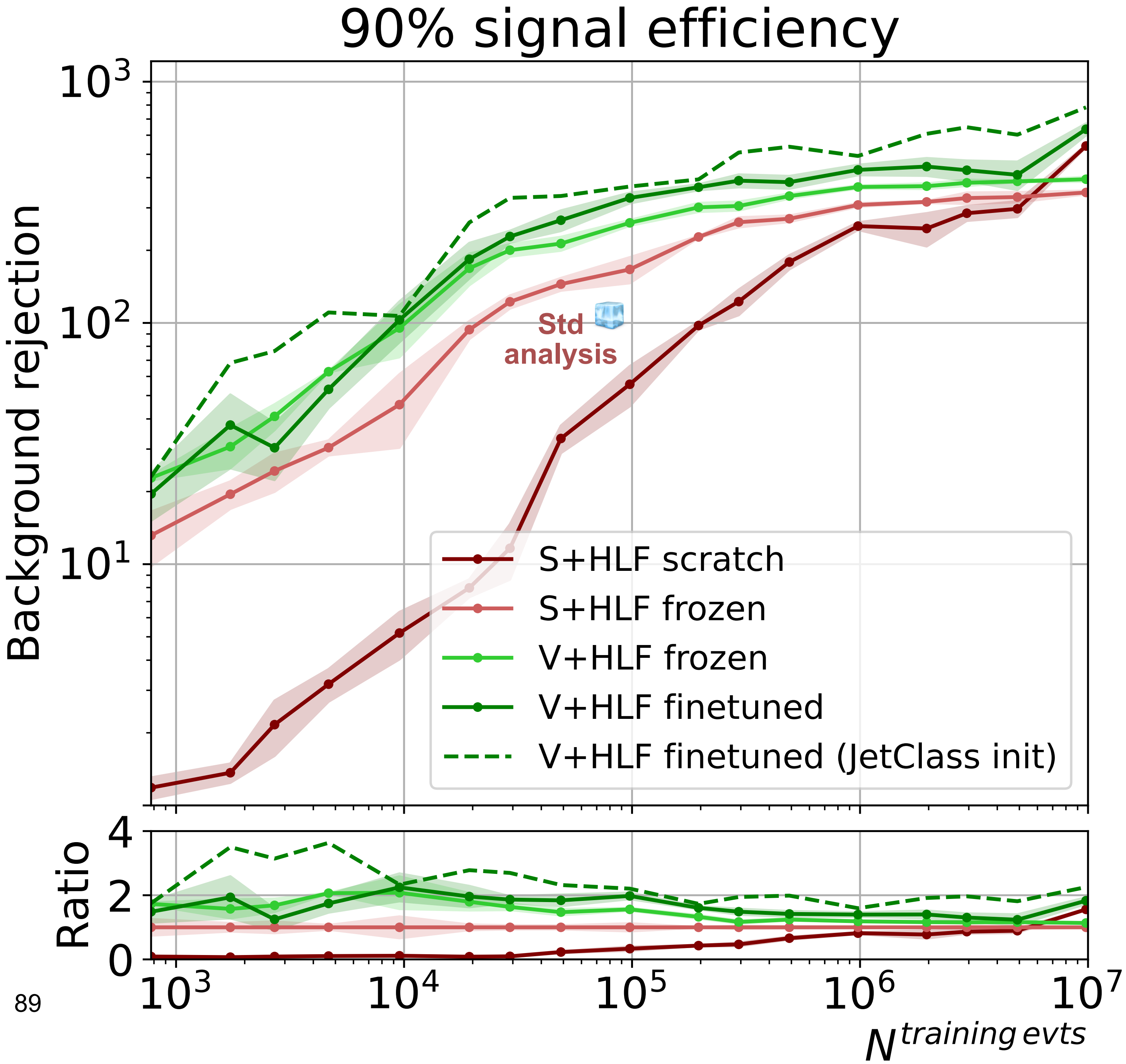
"dumb from-scratch model" eventually surpasses the standard HEP workflow with enough data

Same hierarchy of conclusions for both the vector and vector + HLF trainings backup

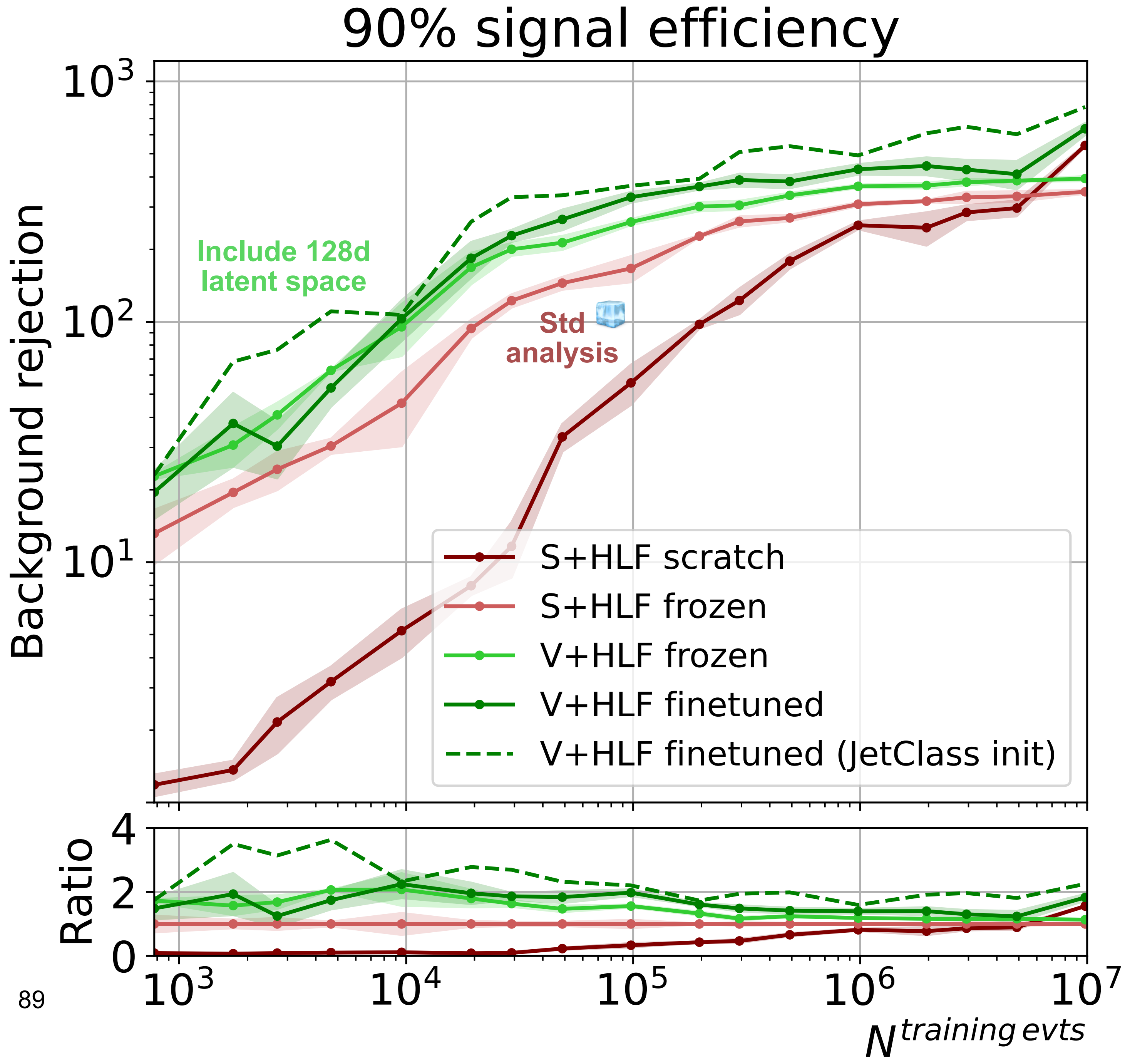
M. Vigl, **NH**, L. Heinrich
2401.13536



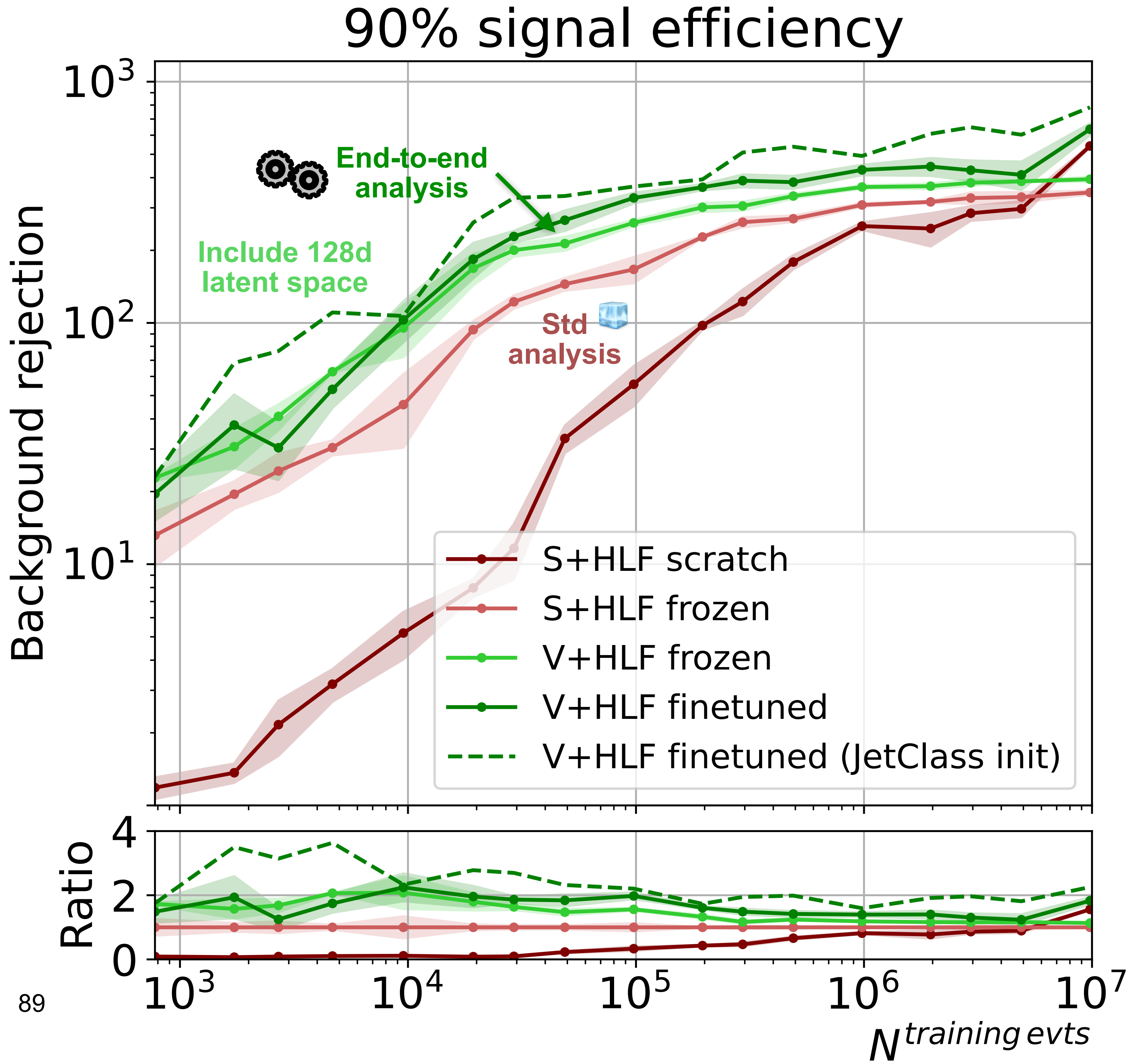
M. Vigl, **NH**, L. Heinrich
2401.13536



M. Vigl, **NH**, L. Heinrich
2401.13536



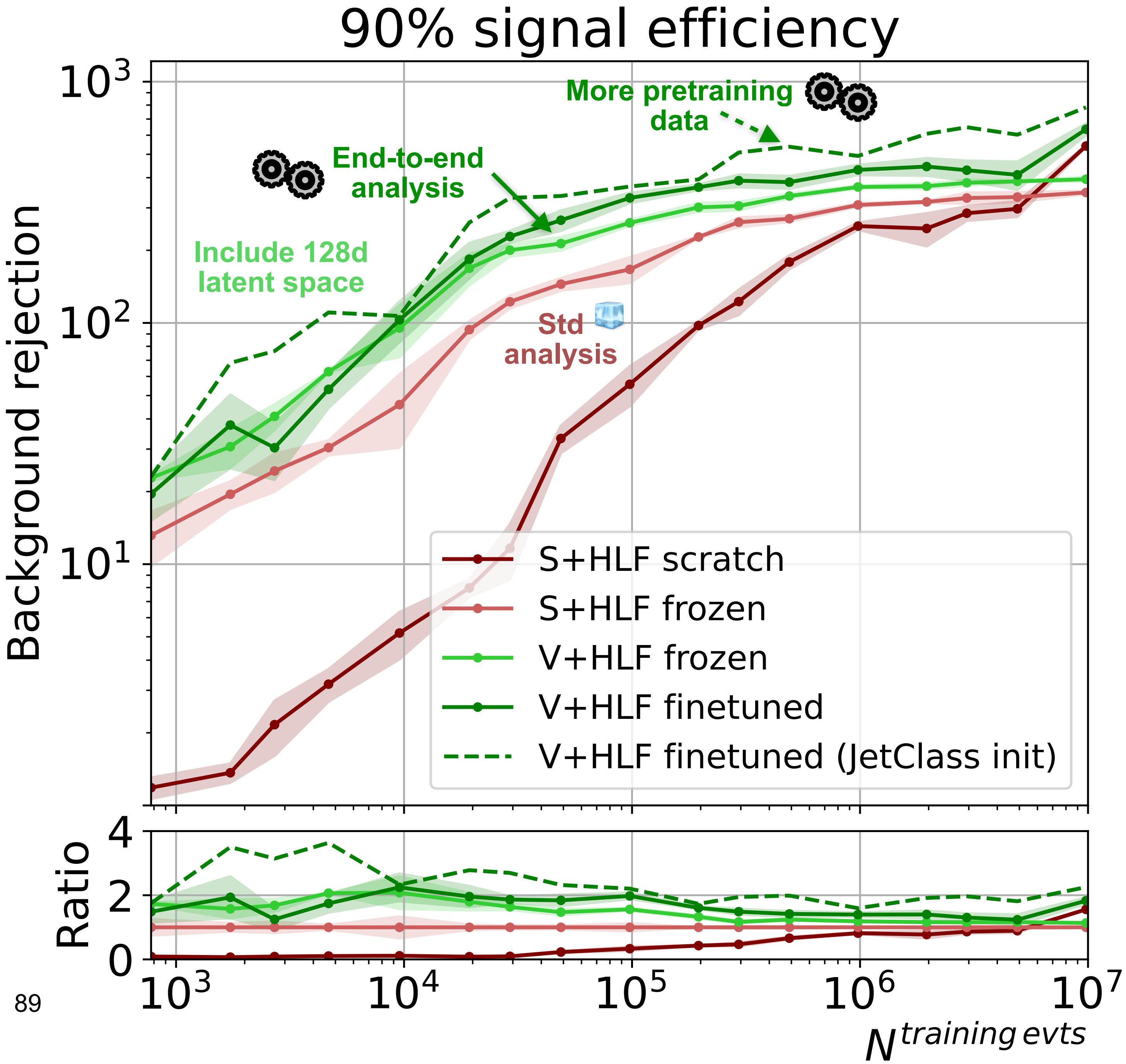
M. Vigl, **NH**, L. Heinrich
2401.13536



Decreases bkg by 2x …

$S/\sqrt{B}$: increases significance by **40%!**

M. Vigl, **NH**, L. Heinrich
2401.13536

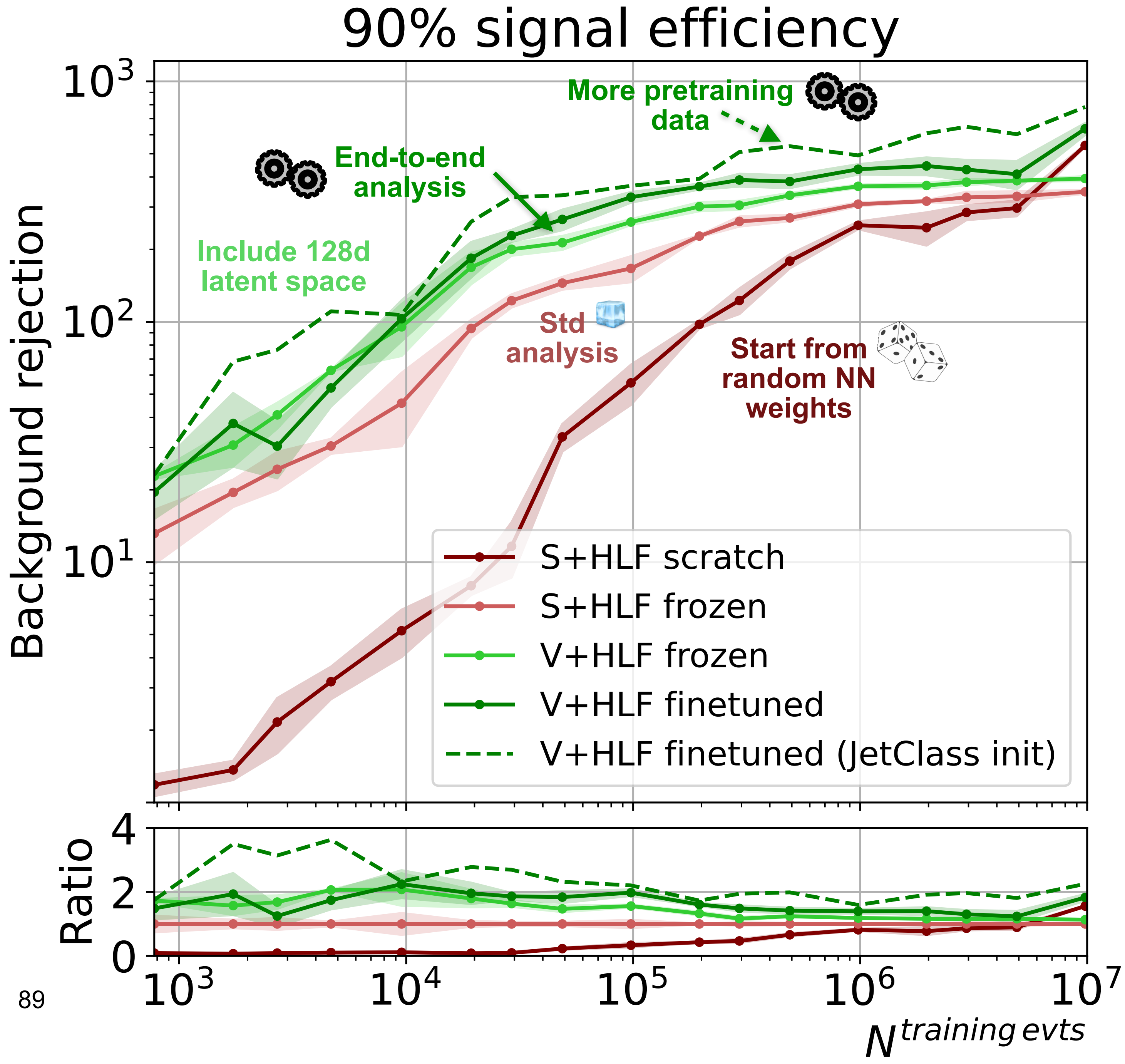


A better Higgs tagger helps analysis performance

Decreases bkg by 2x …

$S/\sqrt{B}$: increases significance by **40%!**

M. Vigl, **NH**, L. Heinrich
2401.13536



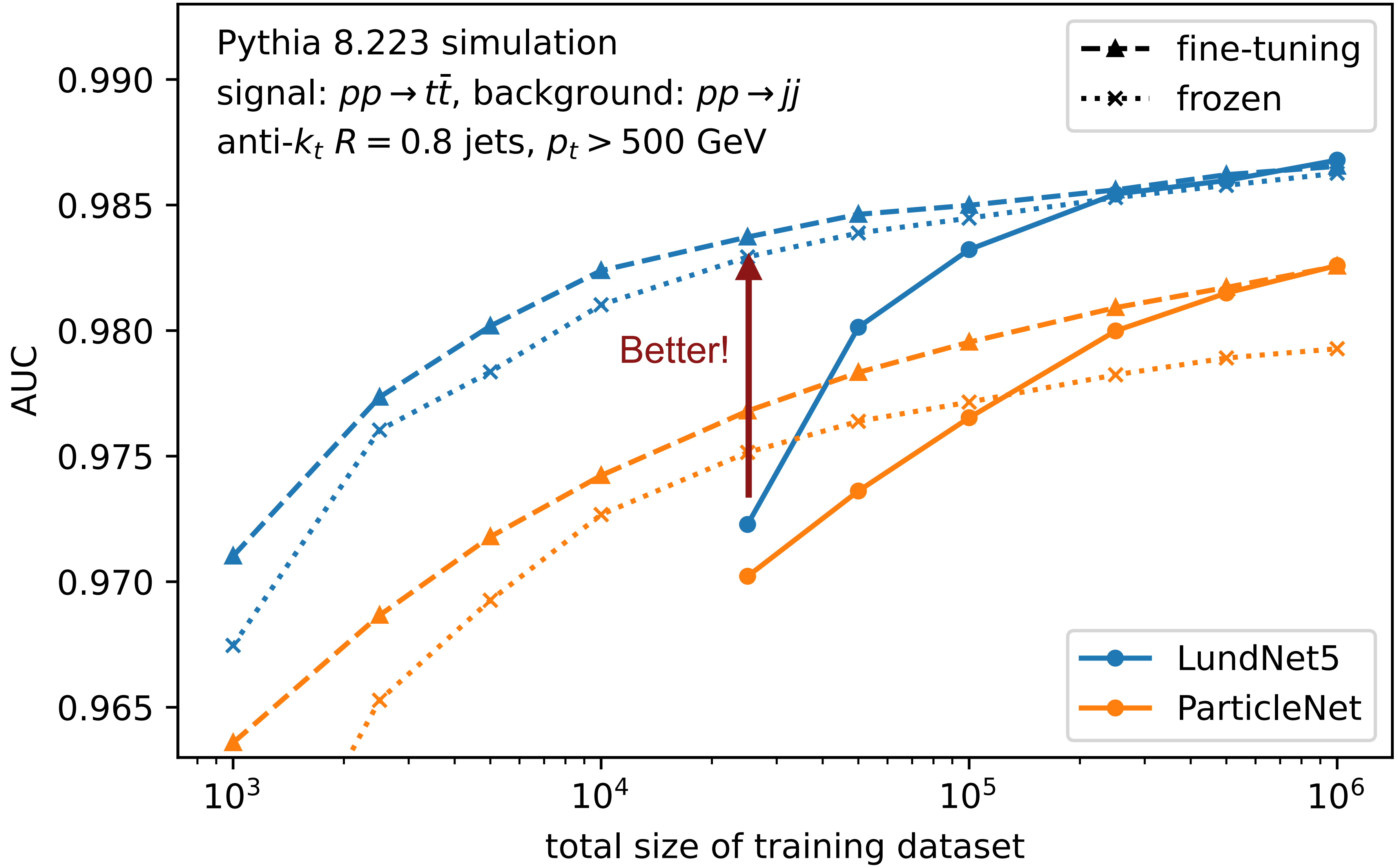
A better Higgs tagger helps analysis performance

But training from scratch, with enough data, will surpass traditional analyses.

Decreases bkg by 2x …

$S/\sqrt{B}$: increases significance by **40%!**

Fine-tuning / Transfer learning



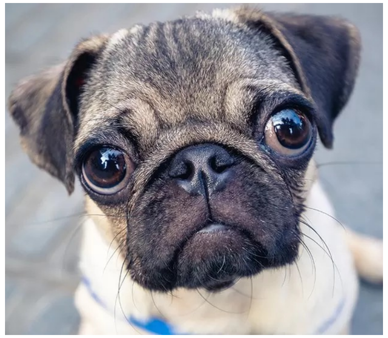
Types of data

Regression $p(y \mid X), y \in \mathbb{R}$ Ex: Housing prices

Classification $p(y \mid X), y \in [\text{class1, class2,} \ldots]$

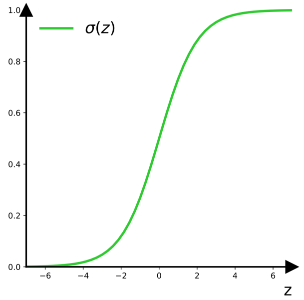
Yesterday: Binary classification

$$y = [\text{cat, dog}]$$



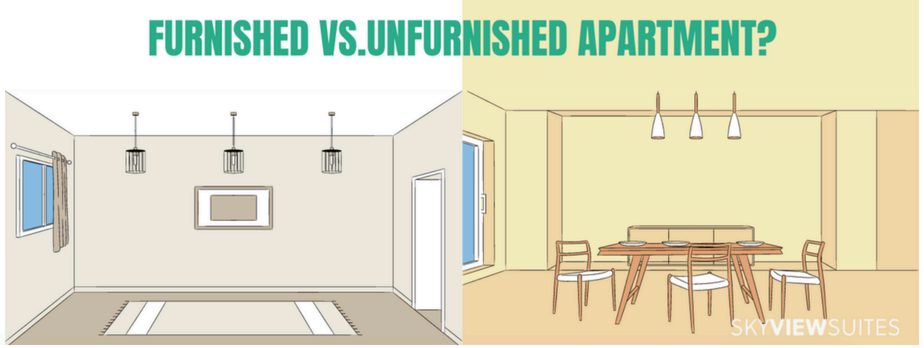
Perceptron:

$$f_\theta(x) = \sigma(\theta^T x)$$



NEXT: Multi-class classification

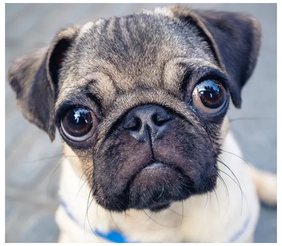
$$y = [\text{not furnished, furnished, semi-furnished}]$$



```python
x_cat = np.zeros_like(df['furnishingstatus'])

for k, case in enumerate(df['furnishingstatus'].unique()):
    print(k,case)

    x_cat[df['furnishingstatus']==case] = k

print(x_cat)
```

```
0 furnished
1 semi-furnished
2 unfurnished
```

Beyond two classes...

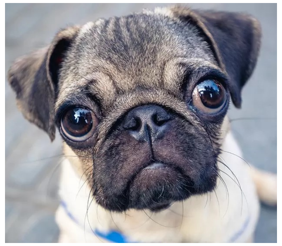


Extending the sigmoid... what if we have more than just cats and dogs?

Targets: One hot vector!

Same idea… more cases!

$$y = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$



$$y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$y = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Specify the non-zero index in a set of classes

Fun fact!!

How you specify words in a dictionary for training language models

$$a = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{aardvark} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \bullet\bullet\bullet \quad \text{exciting} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Prediction: multidimensional outputs…

Linear models

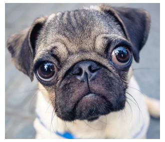
$$z = w^T x, \quad x, w \in \mathbb{R}^d, \quad z \in \mathbb{R}$$

Multi-dimensional linear models

$$z = Wx, \quad x \in \mathbb{R}^d, \quad W \in \mathbb{R}^{K \times d}, \quad z \in \mathbb{R}^K$$

Example: K = 3

z_1: "cat-like"

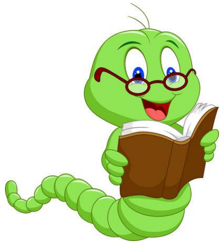


z_2: "dog-like"

z_3: "frog-like"

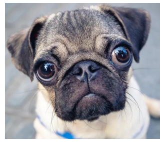
Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$



logits

z_1: "cat-like"



z_2: "dog-like"

z_3: "frog-like"

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

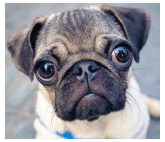
✓

✓

What conditions do I need for p_k to be a probability distribution?

logits

z_1: “cat-like”



z_2: “dog-like”

z_3: “frog-like”

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

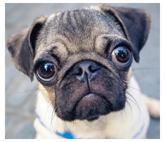
What conditions do I need for p_k to be a probability distribution?

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

logits

z_1: "cat-like"



z_2: "dog-like"

z_3: "frog-like"

Interpreting the output probabilistically

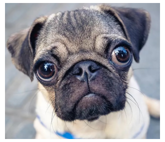
logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

logits


3.2


5.1


-1.7

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

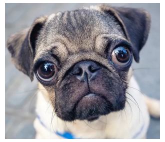
✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

Q: What are some functions that are always positive?

Type your A in chat!

logits

 3.2

 5.1 ⟶

 -1.7

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

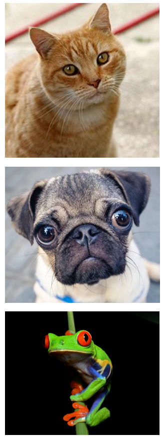
✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

Q: What are some functions that are always positive?

Type your A in chat!

logits



3.2

5.1 ⟶

-1.7

x^2 $\exp$ σ ReLU $|x|$

Interpreting the output probabilistically

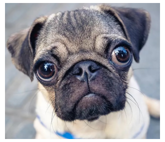
logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

Q: What are some functions that are always positive?

Type your A in chat!

	logits		Unnormalized probabilities
	3.2		24.5
	5.1	$\xrightarrow{\text{exp}}$	164.0
	-1.7		0.18

x^2

$\exp$

σ

ReLU

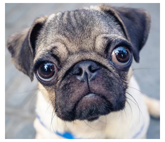
$|x|$

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

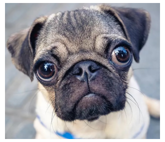
	logits		Unnormalized probabilities
	3.2		24.5
	5.1	$\xrightarrow{\exp}$	164.0
	-1.7		0.18

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

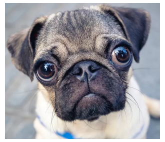
	logits		Unnormalized probabilities	
	3.2		24.5	
	5.1	exp →	164.0	→
	-1.7		0.18	

Interpreting the output probabilistically

logits: $z = Wx, \quad z \in \mathbb{R}^K$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

	logits		Unnormalized probabilities		Class probabilities
	3.2		24.5		0.13
	5.1	exp →	164.0	normalize →	0.87
	-1.7		0.18		0.00

Interpreting the output probabilistically

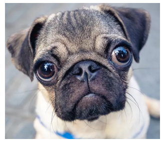
logits: $z = Wx, \quad z \in \mathbb{R}^K$

Softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

✓ Positivity $p_i > 0$

✓ Sums to unity: $\sum_{i=1}^{K} p_i = 1$

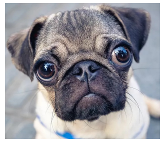
	logits		Unnormalized probabilities		Class probabilities
	3.2		24.5		0.13
	5.1	exp →	164.0	normalize →	0.87
	-1.7		0.18		0.00

Loss function: Cross entropy

logits: $z = Wx, \quad z \in \mathbb{R}^K$

Softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

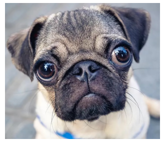
	logits		Class probabilities
	3.2		0.13
	5.1	Softmax →	0.87
	-1.7		0.00

Loss function: Cross entropy

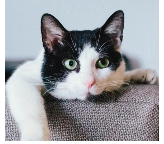
logits: $z = Wx, \quad z \in \mathbb{R}^K$

Softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

	logits		Class probabilities	Target
	3.2		0.13	1
	5.1	Softmax →	0.87	0
	-1.7		0.00	0

$= y$ (Correct probabilities)

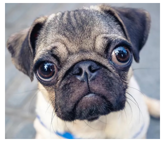
 = x

Loss function: Cross entropy

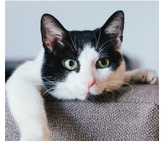
logits: $z = Wx, \quad z \in \mathbb{R}^K$

Softmax function:

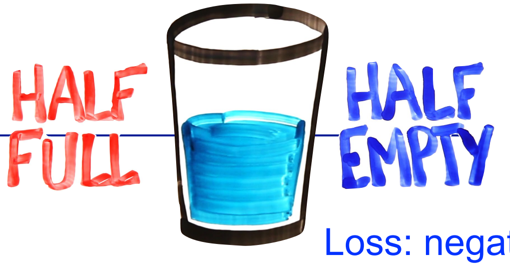
$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

	logits		Class probabilities		Target
	3.2		0.13		1
	5.1	Softmax →	0.87	← Compare probability distributions →	0 = y
	-1.7		0.00		0

Correct probabilities

 = x

Loss function: Cross entropy

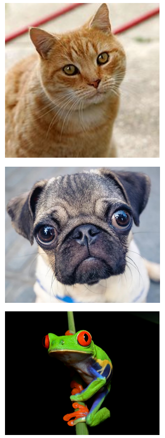


Loss: negative log likelihood

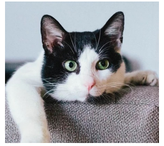
logits: $z = Wx, \quad z \in \mathbb{R}^K$

Softmax function:

$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

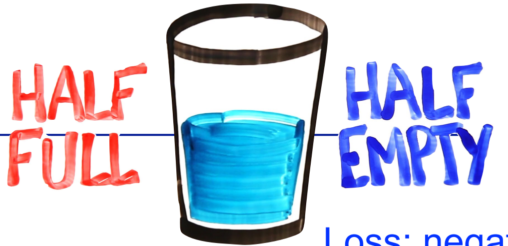


	logits		Class probabilities		Target
	3.2		0.13		1
	5.1	Softmax →	0.87	Compare probability distributions ↔	0 = y
	-1.7		0.00		0


= x

Correct probabilities

Loss function: Cross entropy



logits: $z = Wx, \quad z \in \mathbb{R}^K$

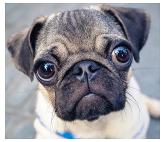
Softmax function:

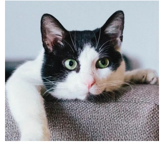
$$p_i = \frac{\exp(z_i)}{\sum_{i=1}^{K} \exp(z_i)}$$

Cross entropy loss

Loss: negative log likelihood

$$\mathcal{L} = -\log P(Y = y_i \,|\, X = x)$$
$$= -\log \frac{\exp(z_{y_i})}{\sum_j \exp(z_j)}$$

	logits		Class probabilities		Target
	3.2		0.13	Compare probability distributions	1
	5.1	Softmax →	0.87	⟷	0 = y
	-1.7		0.00		0

 = x

Correct probabilities