Mastering Model Building

How to design (and debug) your ML model

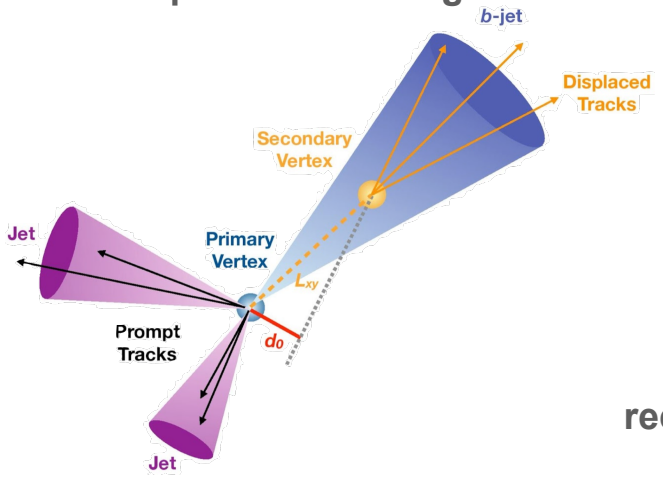
Nicole Hartman
nicole.hartman@tum.de

Train the Trainer workshop
7th February 2023

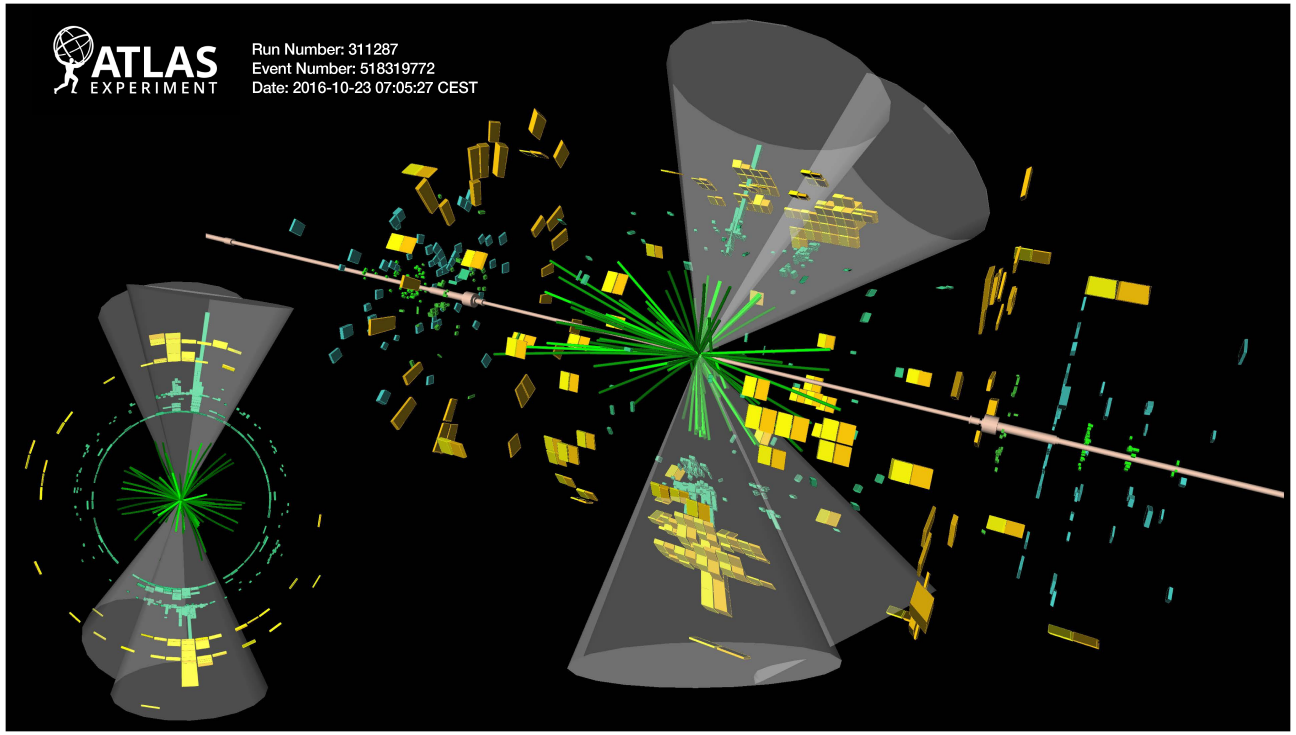
Who am I?

- ORIGINS Data Science Lab post-doc
- Particle physicist 
- Big data -> big opportunities!!

Set and sequence modeling



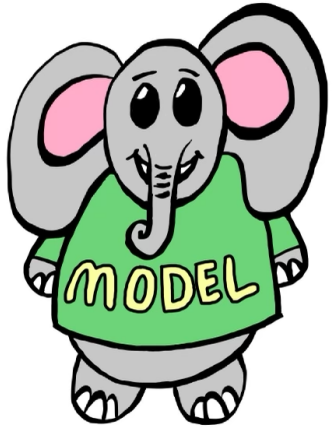
Event reconstruction

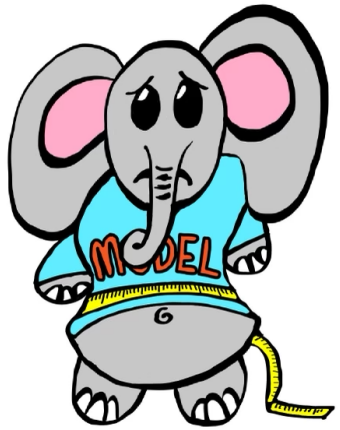


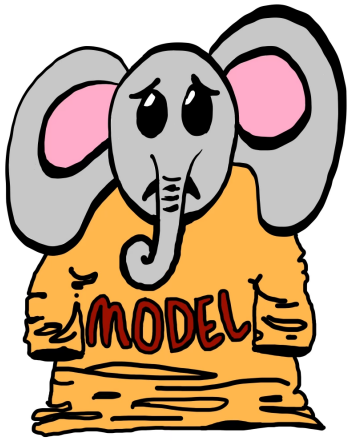
Generative models / Density Ratio Estimation

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







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

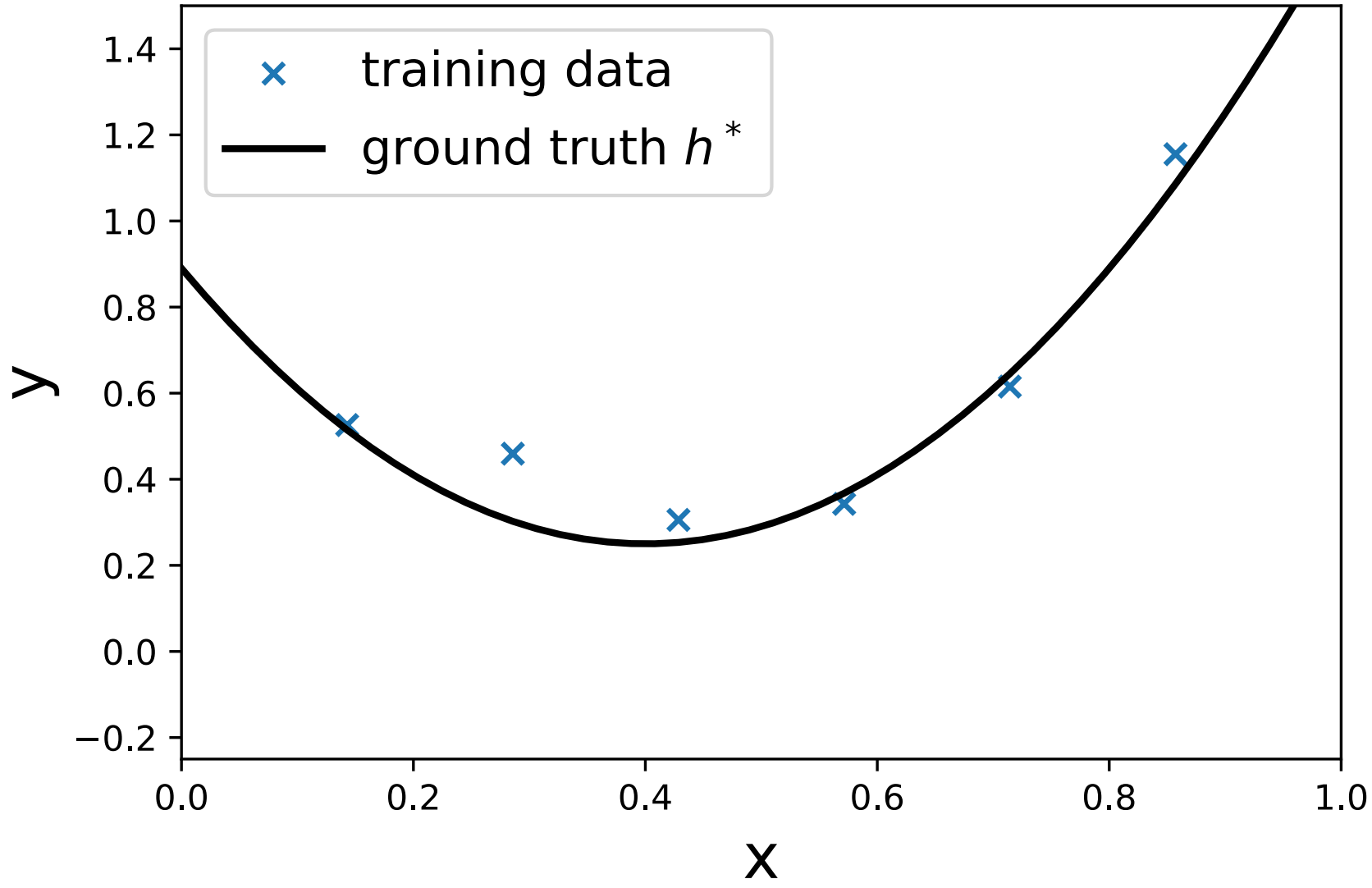
Training techniques

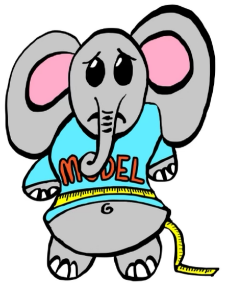
Statistical learning theory

Bias / variance trade-off

Would love feedback + discussions!

Working example

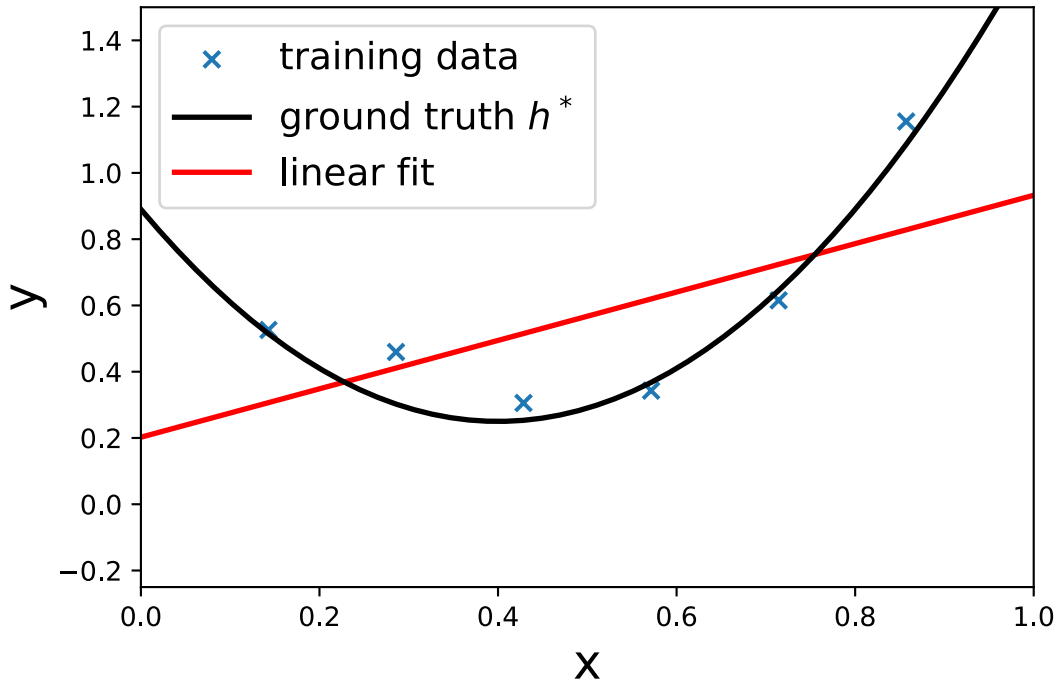




Underfitting

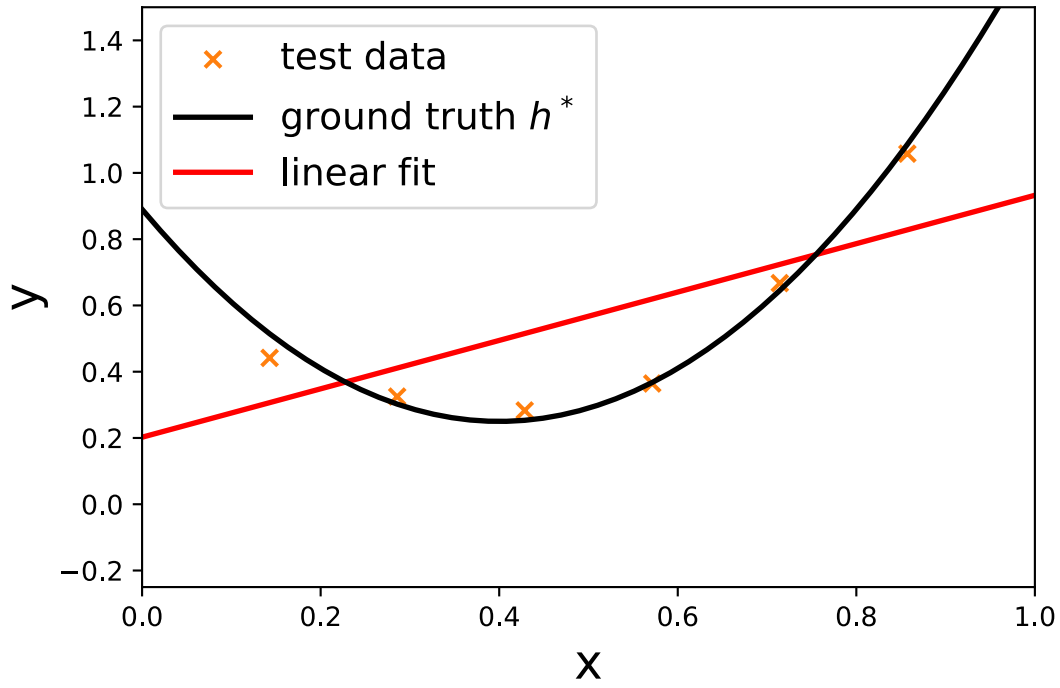
Model is not expressive enough

Training data

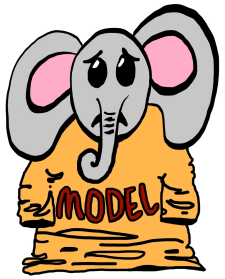


High training error

Test data



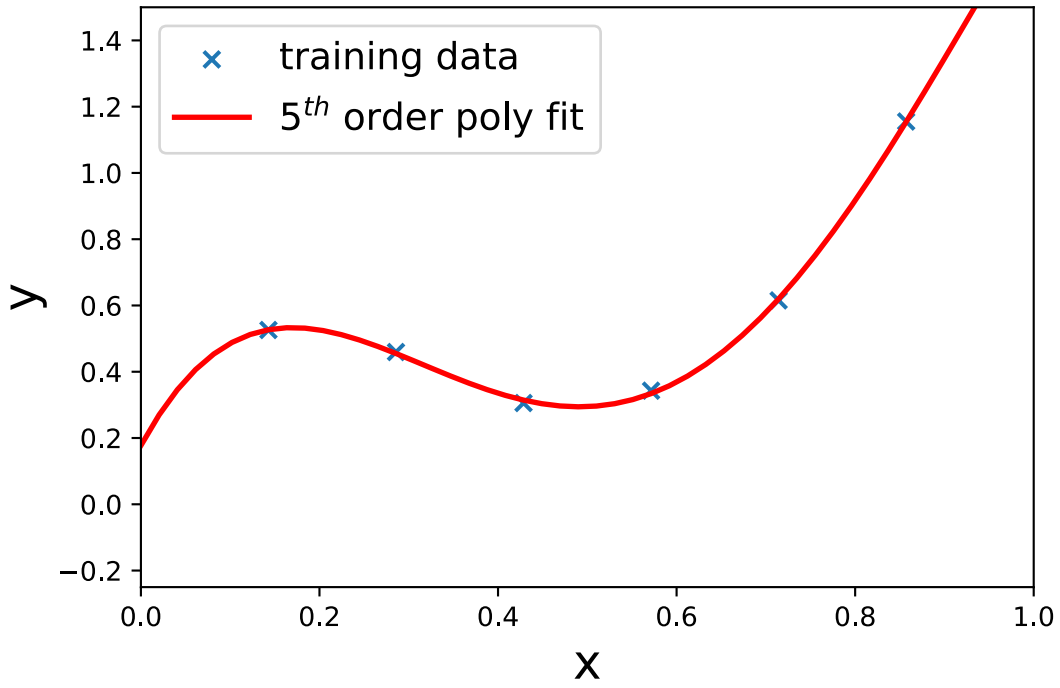
Also high test error



Overfitting

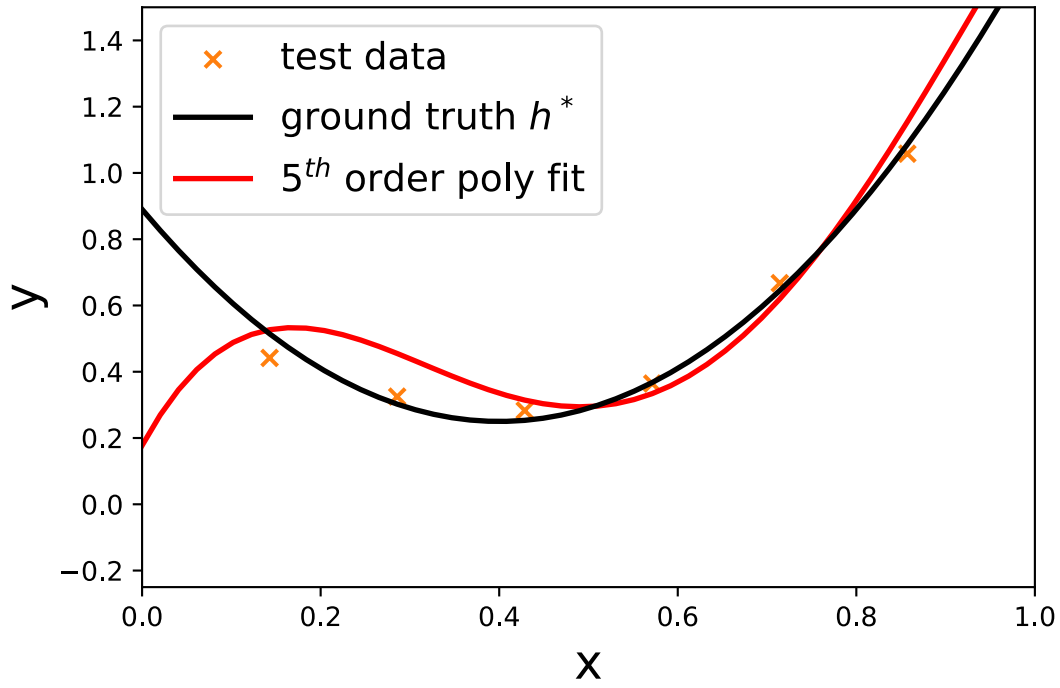
Model too expressive to generalize to unseen dataset

Training data

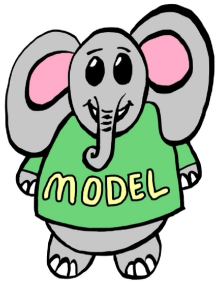


Small (zero) training error

Test data



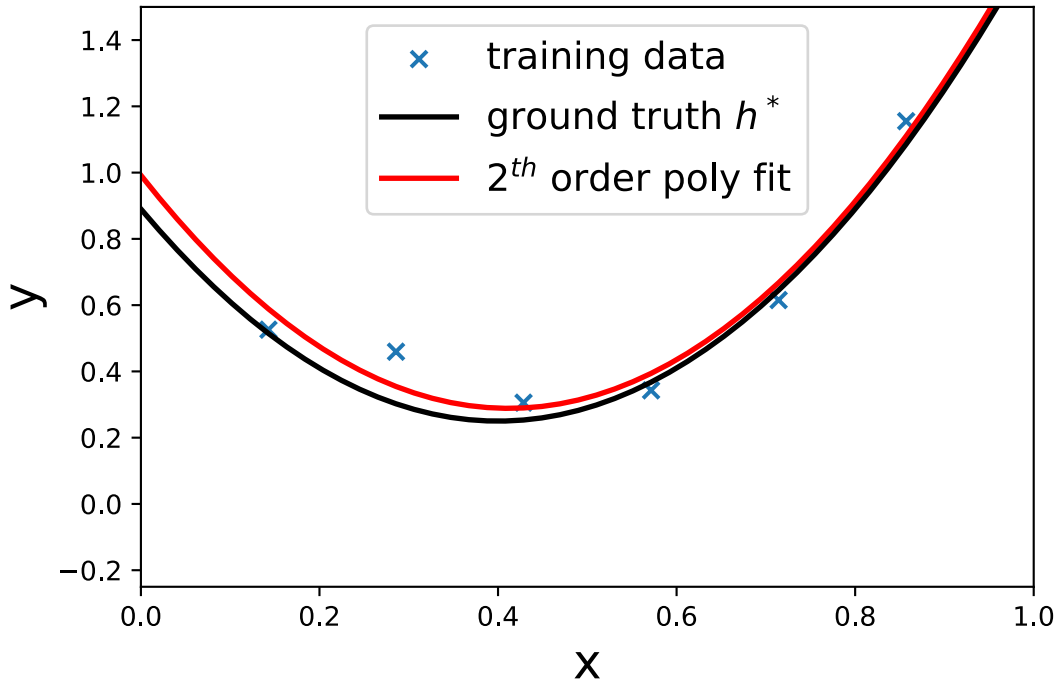
High test error



Optimal model complexity

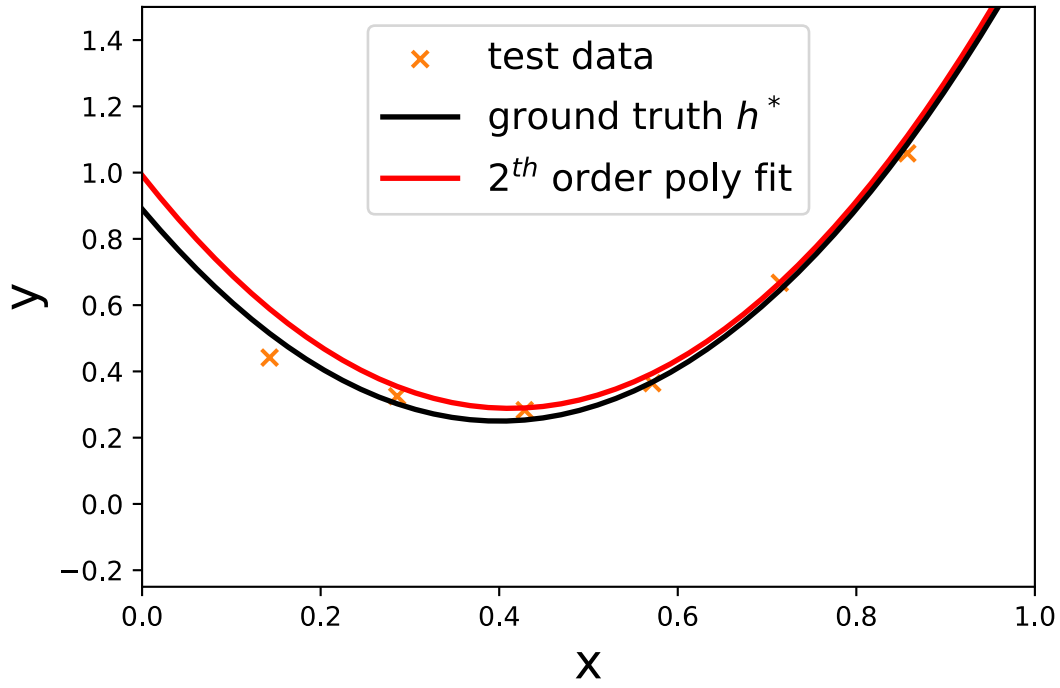
Fit 2nd order polynomial to quadratic distribution

Training data



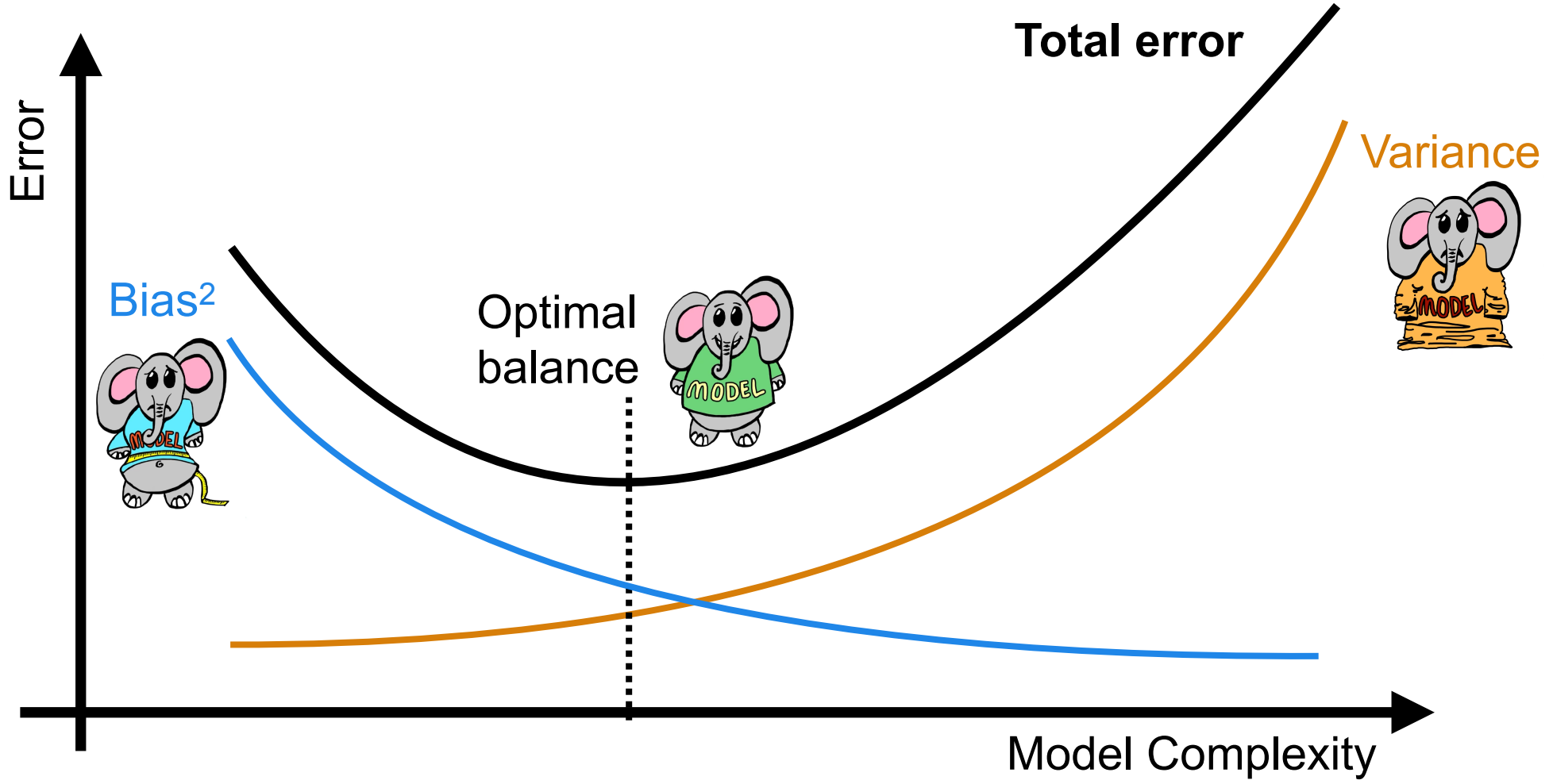
Small training error

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 $\hat{h}_S$ on dataset S
- Consider test point (x,y) and quantify the *expected test error*:

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

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

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

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
- Substitute this back into the MSE expression

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

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

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

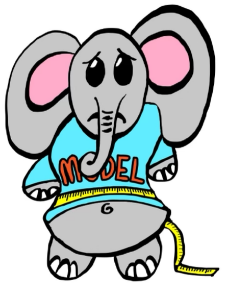
No cross-term because $\mathbb{E}\left[h_{avg}(x) - h_S(x)\right] = 0$

Bias²

Error on this class of models

Variance

How does this instantiation compare with the other possible ones?



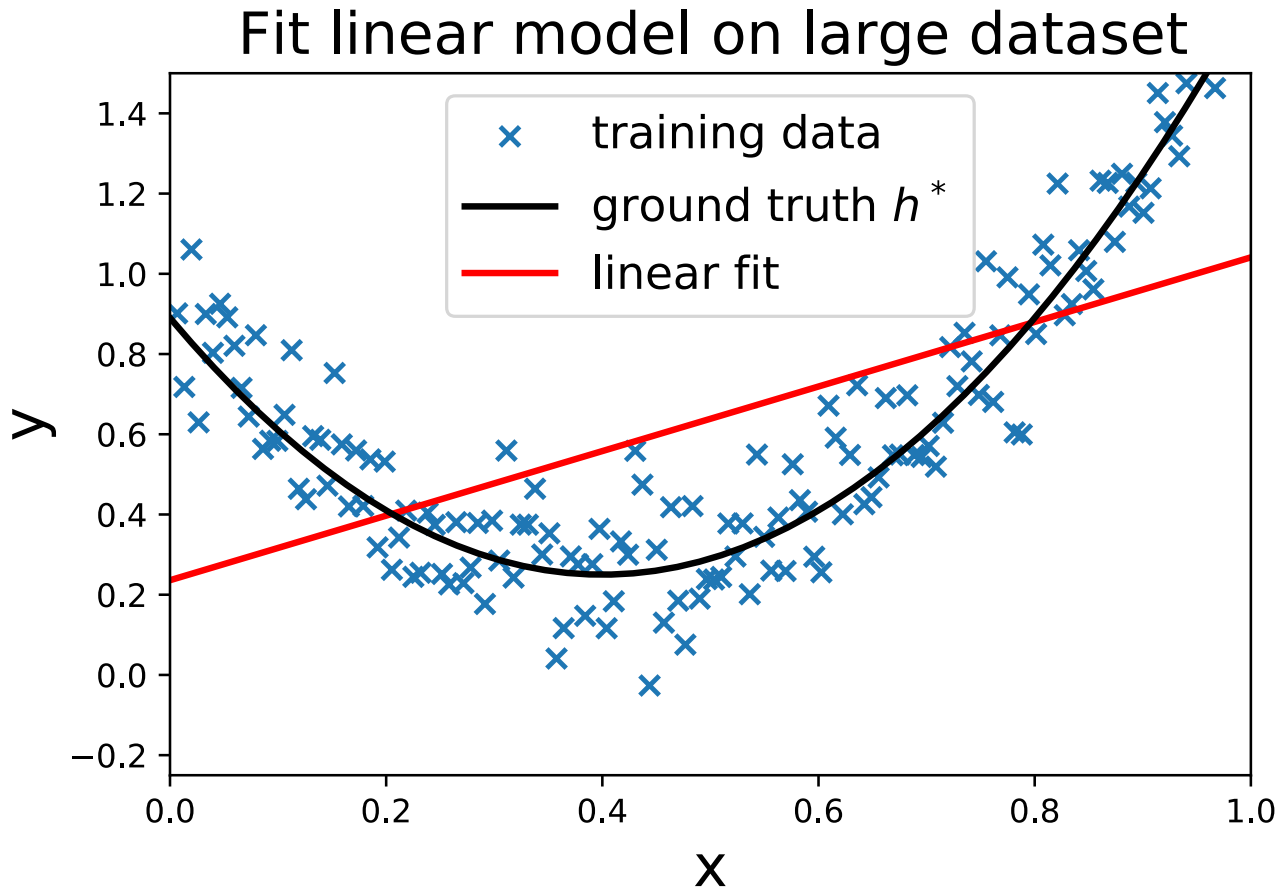
High bias: diagnostics

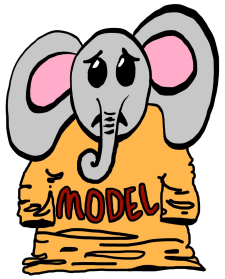
$$MSE(x) = \underbrace{(h^*(x) - h_{avg}(x))^2}_{\text{Bias}^2} + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

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

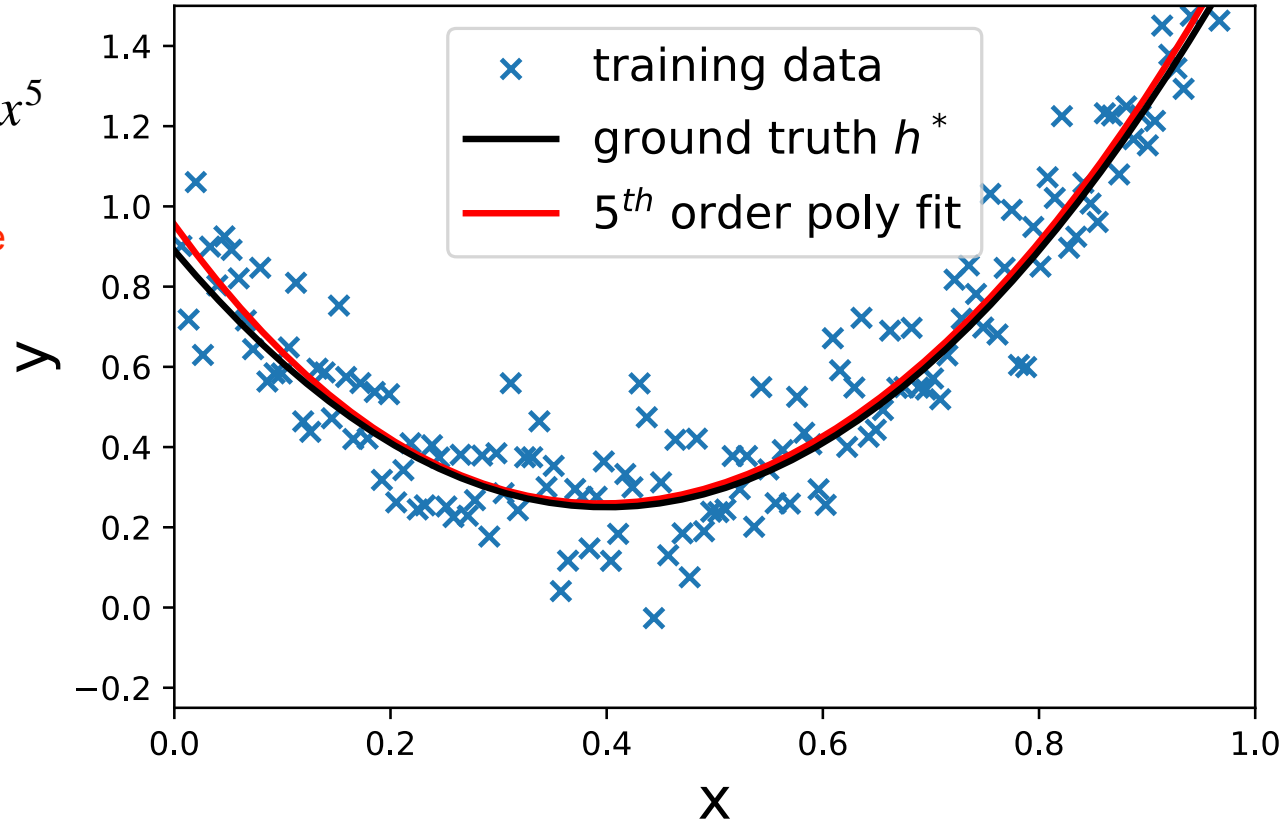
5th order polynomial fit

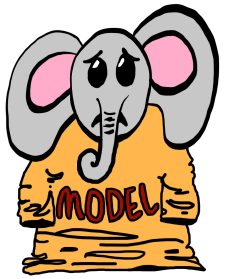
$$h_S(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5$$

Can learn to set these coefficients to 0

The training error decreases as we increase the training data.

Fit 5^{th} order polynomial on large dataset



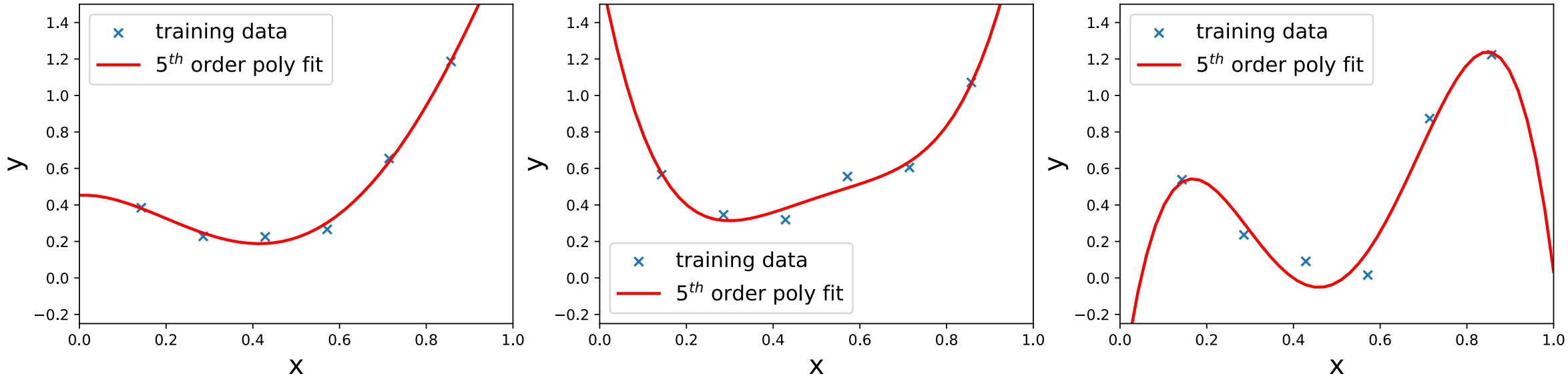


High variance: intuition

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

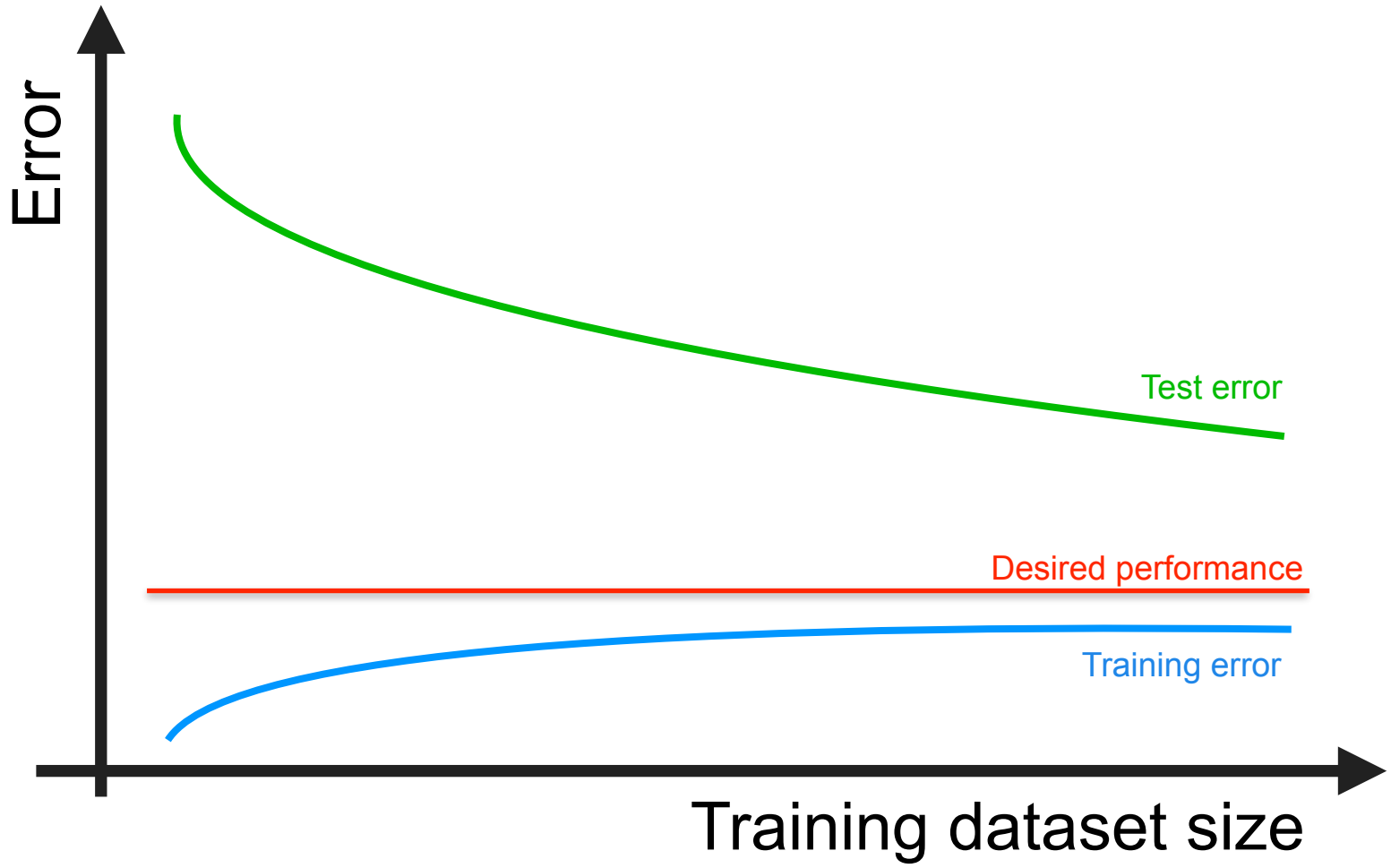
Variance

fitting 5^{th} degree polynomial on different datasets



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

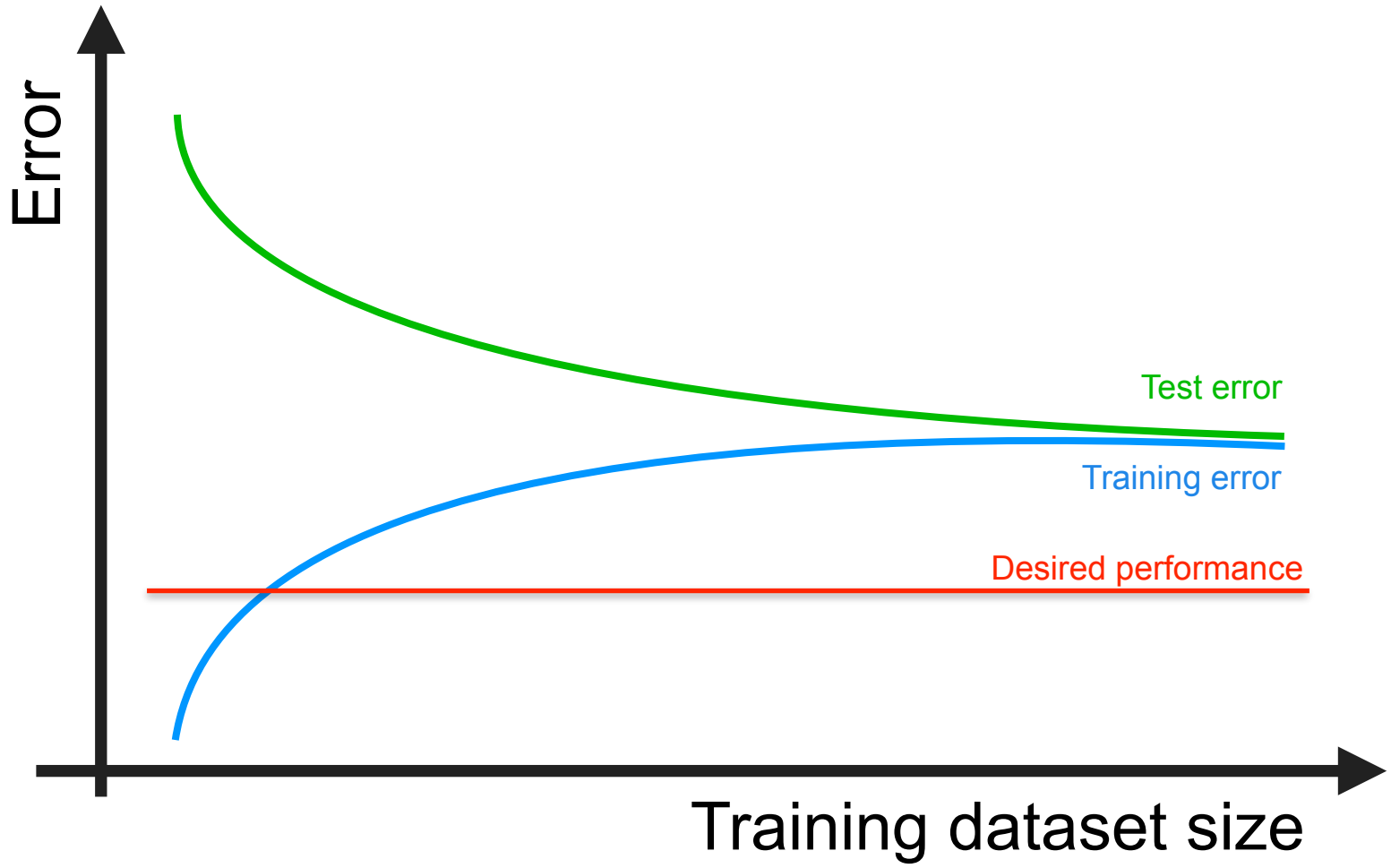
What's the culprit?



High bias?

High Variance?

What's the culprit?



High bias?

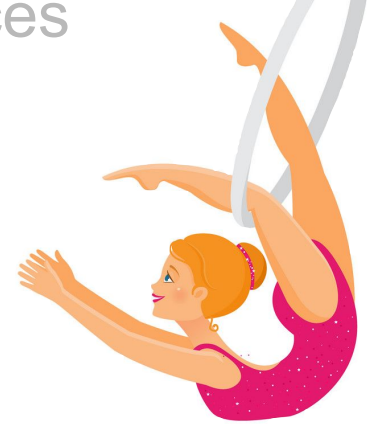
High Variance?

Starting off…

Open question

Feature choices

Going deeper



Training techniques

Statistical learning theory

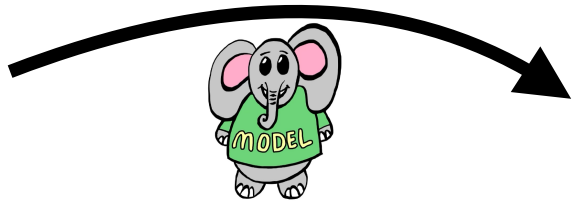
Bias / variance trade-off

Would love feedback + discussions!

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.

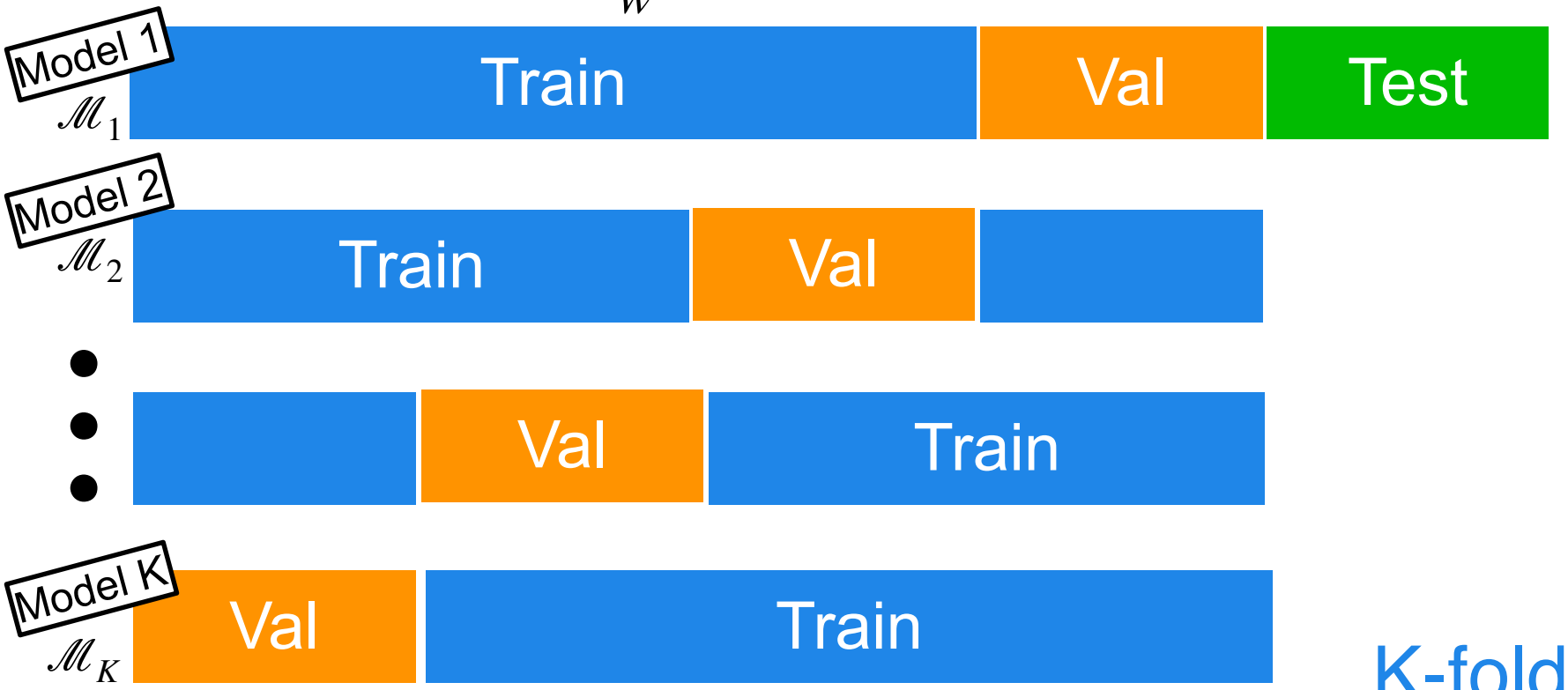
Want a *large* training dataset to minimize $\mathscr{L}$.

Image by brgfx on Freepik

How to maximize statistics?

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

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



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

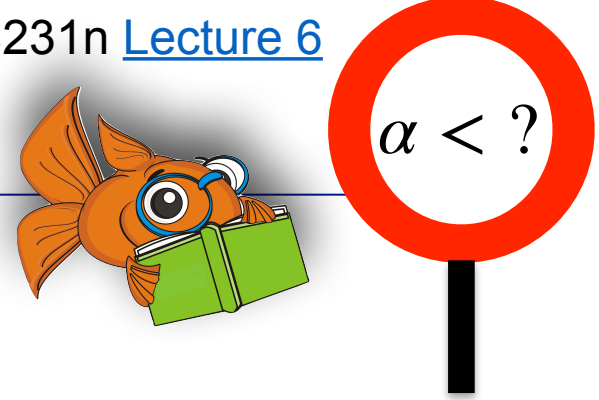
$$\mathscr{M} = \frac{1}{K} \sum_{i=1}^{K} \mathscr{M}_i$$

Ensembling helps performance!

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

K-fold cross validation

Learning rate



Minimize $\mathscr{L}$ by SGD

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

How to choose α?

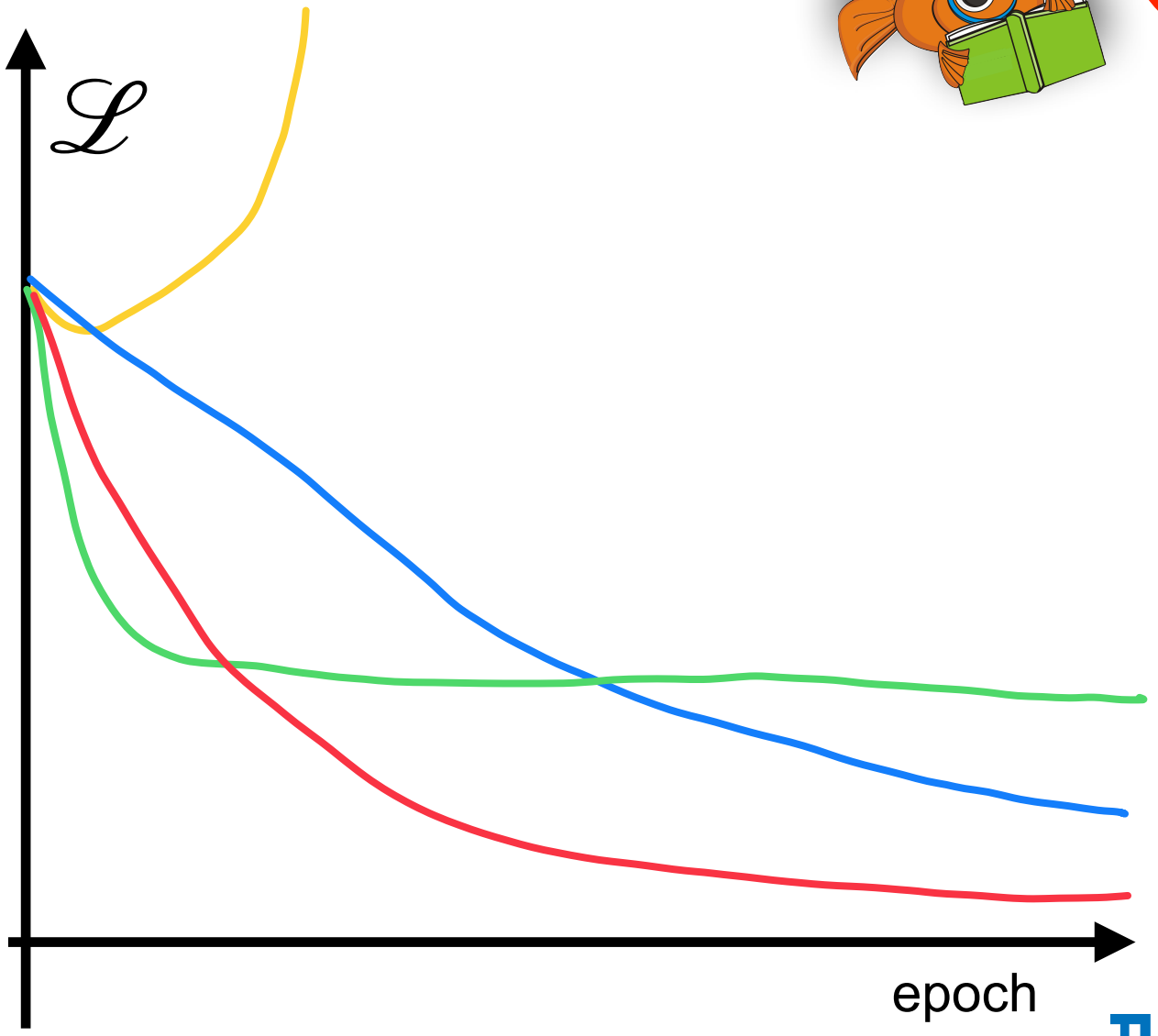
Label the loss curves!

very high learning rate

high learning rate

good learning rate

low learning rate



Batch size

Minimize $\mathscr{L}$ by SGD

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

Make a MC estimate

m: mini-batch size

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

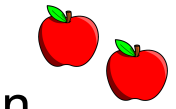
  E.g, 256, 512, 1024

Andrej Karpathy post
@karpathy

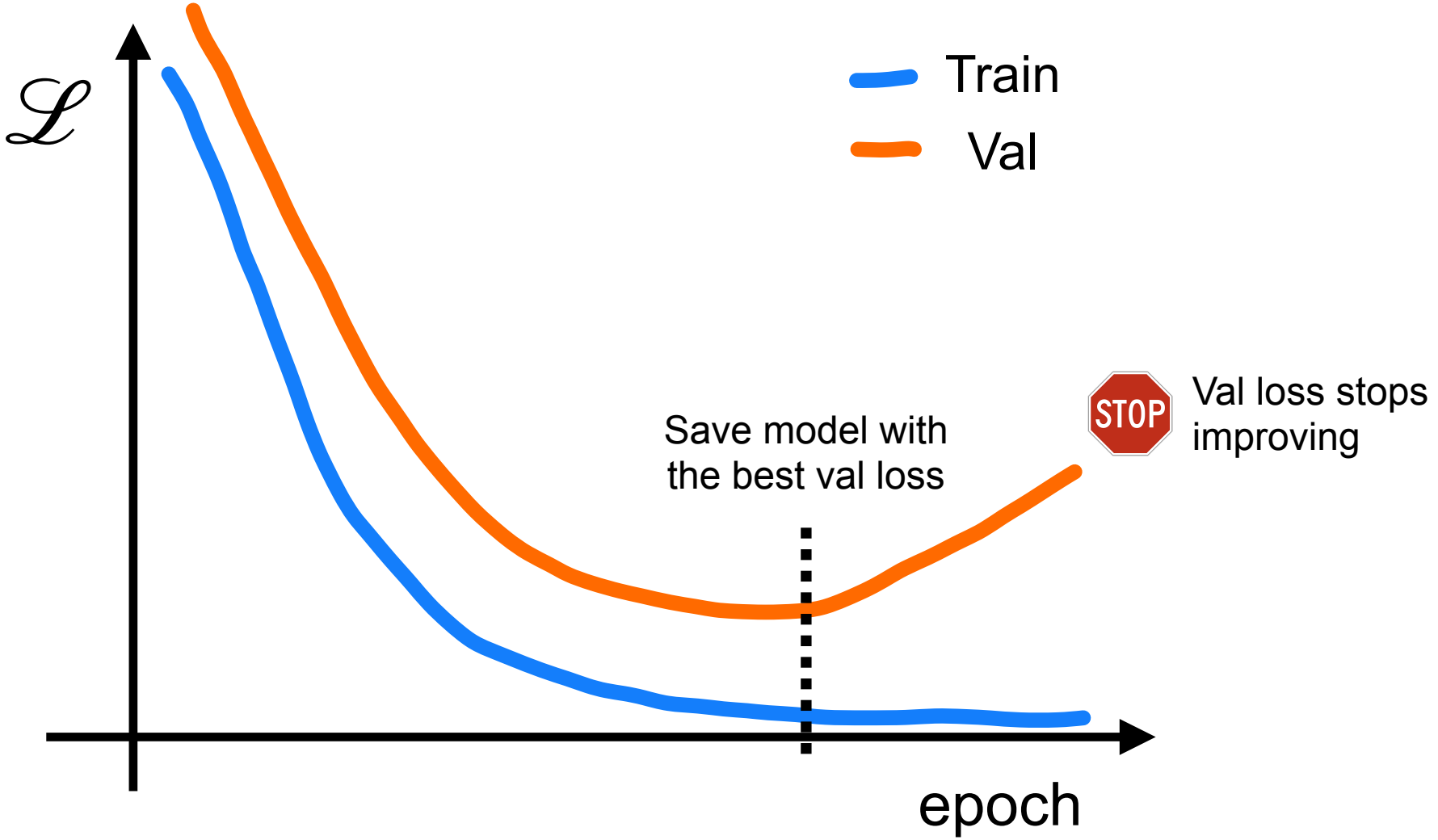
The most dramatic optimization to nanoGPT so far (~25% speedup) is to simply increase vocab size from 50257 to 50304 (nearest multiple of 64). This calculates added useless dimensions but goes down a different kernel path with much higher occupancy. Careful with your Powers of 2.

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



Hyperparameter search

Already many options…

1 Activations

Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

Leaky ReLU $\max(0.1x, x)$

tanh $\tanh(x)$

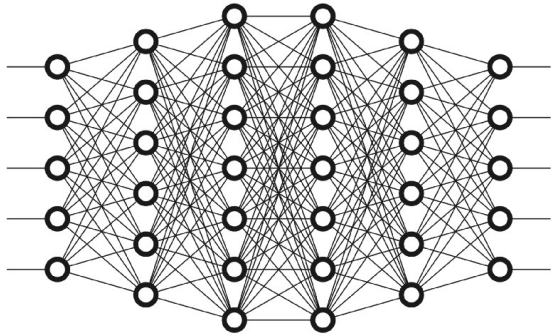
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU $\max(0, x)$

ELU $\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

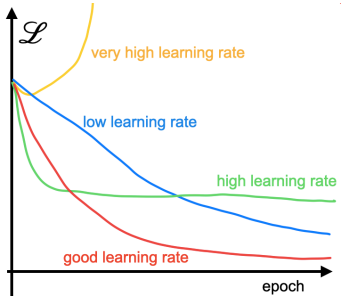
Medium article

2 Number of layers



3 Nodes / layer

4 Learning rate



+ many more!

in this talk and others!

Coarse scan:

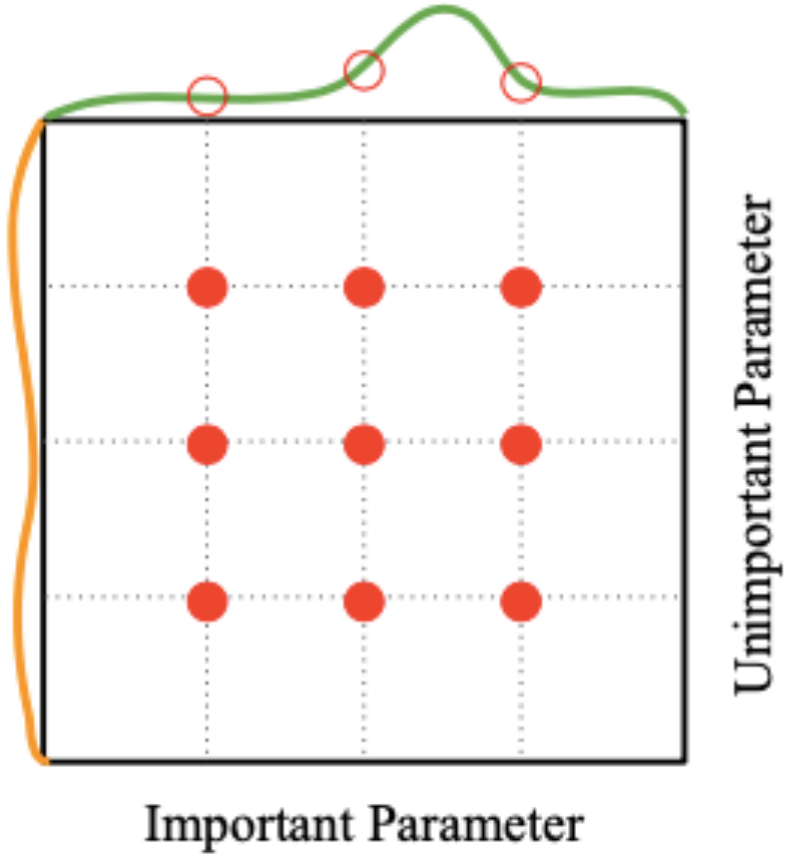
3 activations {sigmoid, ReLU, ELU}
x **3** layers {5, 10, 20}
x **4** nodes {10, 50, 250, 200}
x **4** learning rates {1e-2, 3e-3, 1e-3, 3e-4}
x **2** {with and w/o scheduler}

= 384 trainings !!!

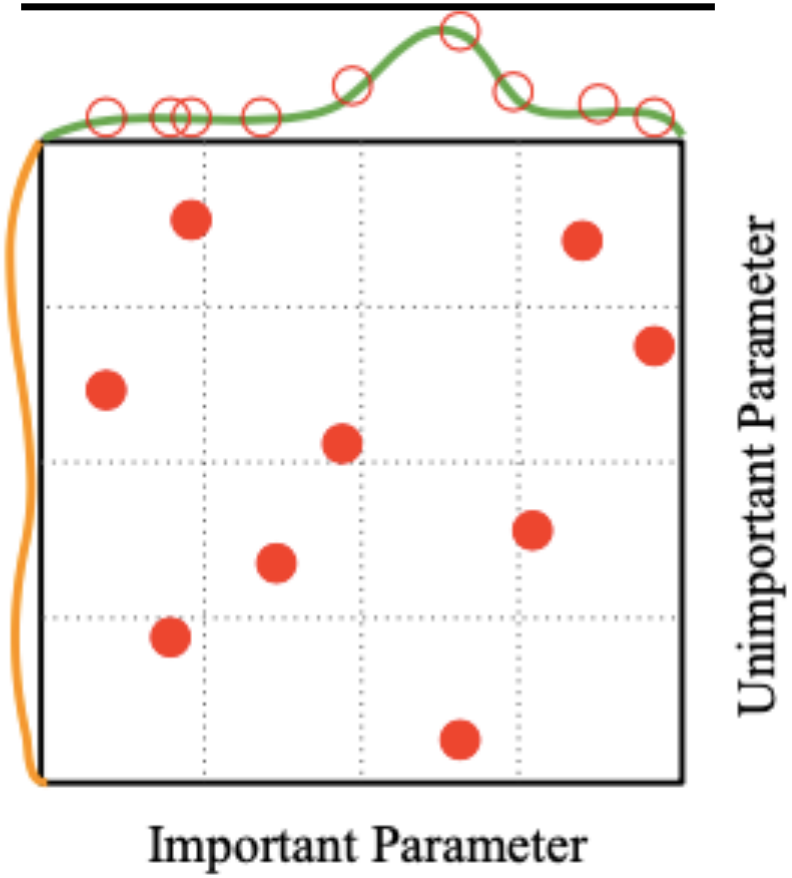
Random Search for Hyper-Parameter Optimization Bergstra and Bengio, 2012
Image from CS231n lecture

Hyper-parameter search

Grid Search

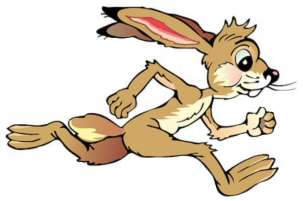


Random Search



Not all hyper-parameters are equal!

Hyperparameter strategies



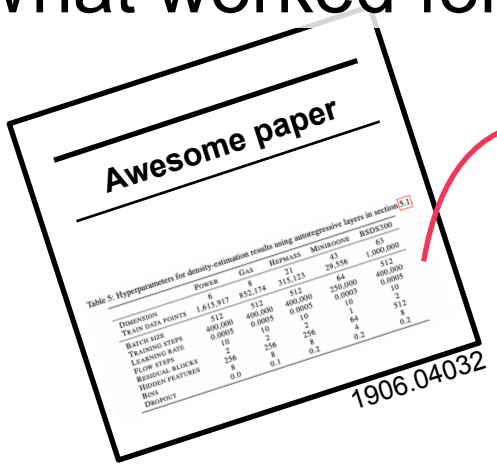
Fast prototyping

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

Can automate!

OPTUNA

What worked for others?



Starting point random search

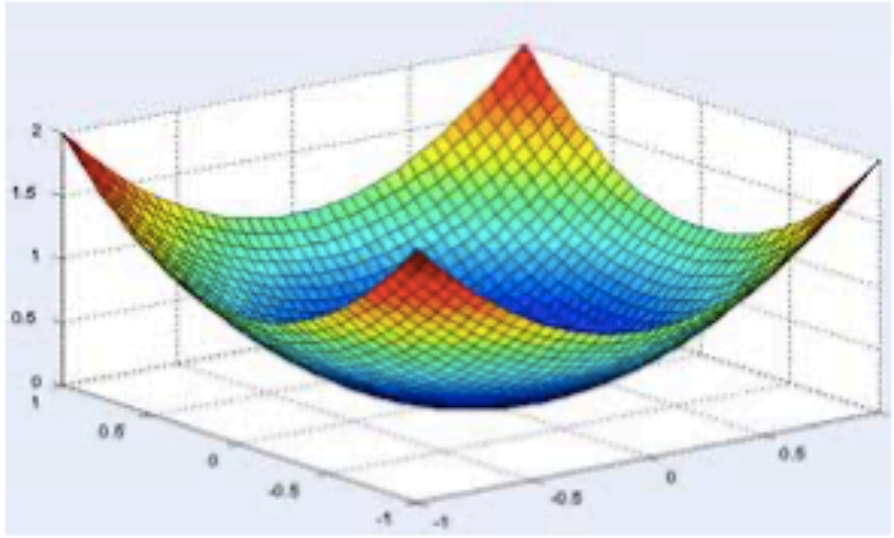
Scan in log space

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 !!!

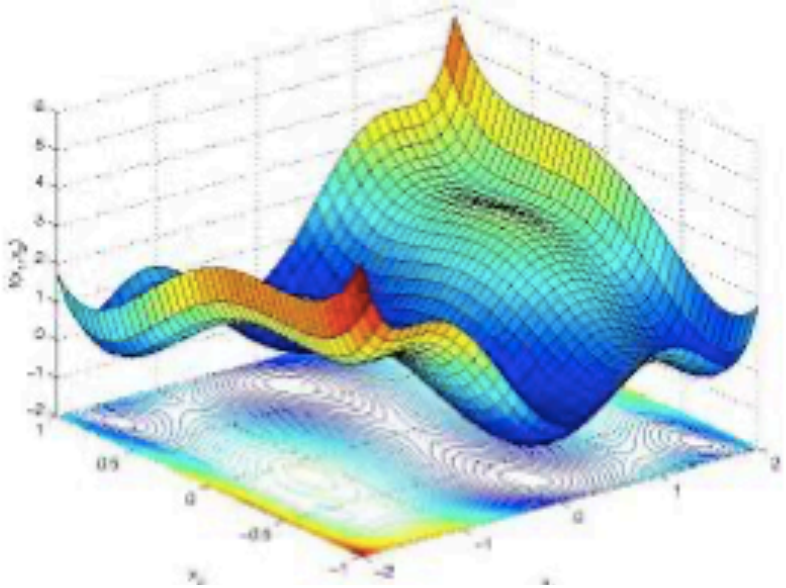
Loss landscape

Linear functions



(convex loss)

Neural networks



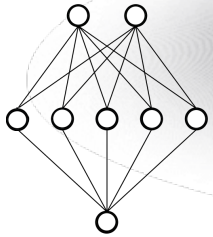
(non-convex)

Starting off…

Open question

Feature choices

Going deeper



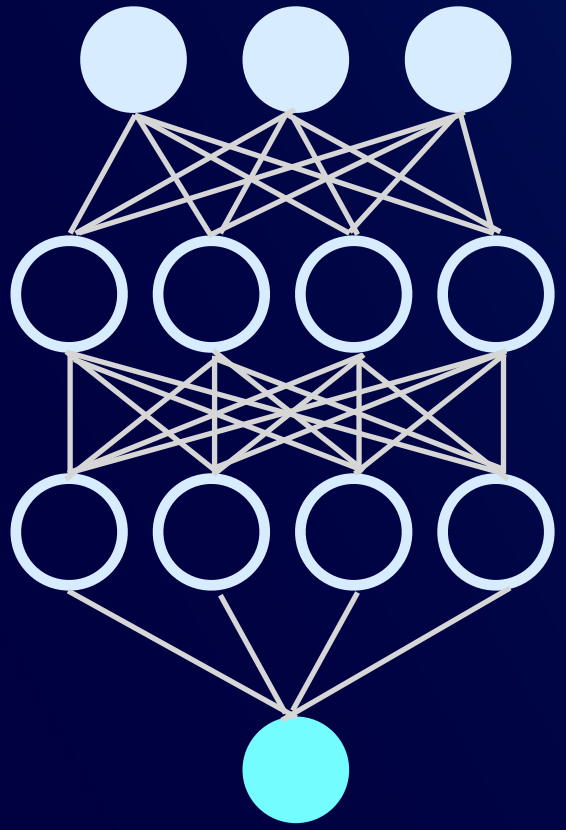
Training techniques

Statistical learning theory

Bias / variance trade-off

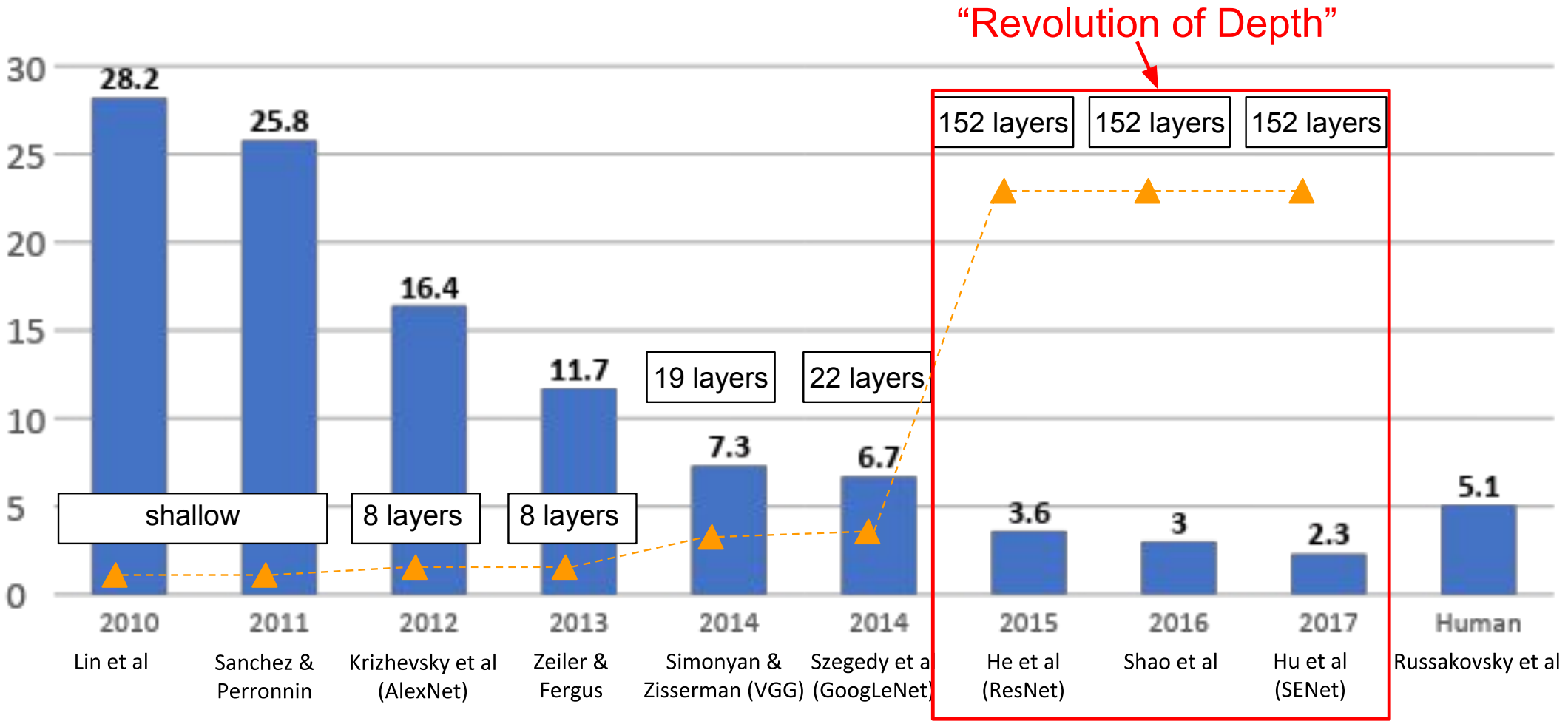
Would love feedback + discussions!

Models in the Deep Learning Era

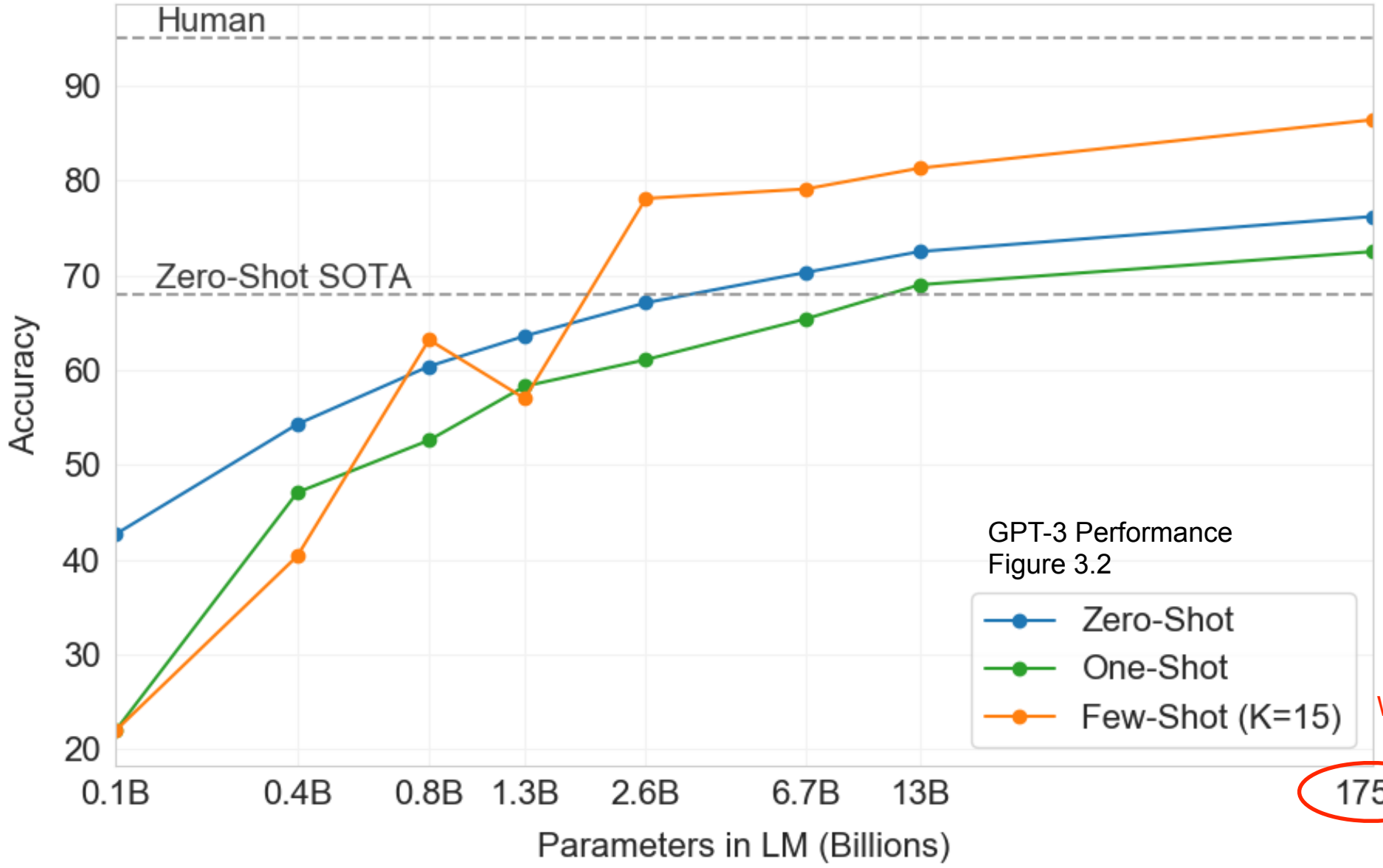


The Deep Learning Revolution

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.

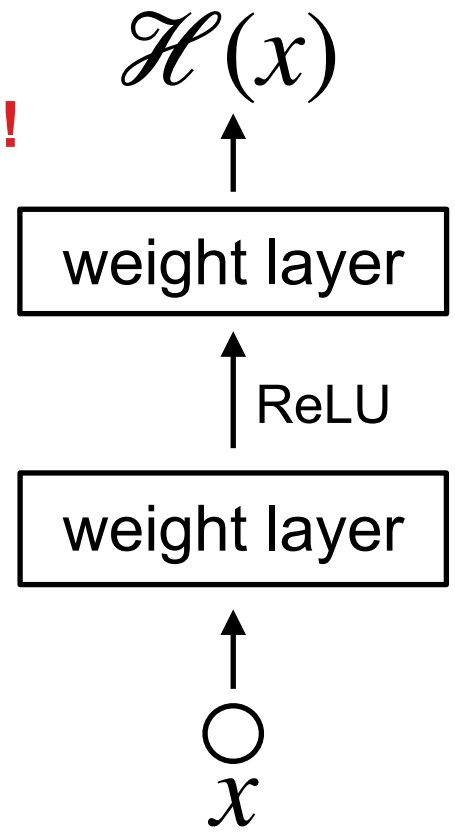
Deep Learning philosophy…

Bigger = better

ResNet building block

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

Degrades performance!



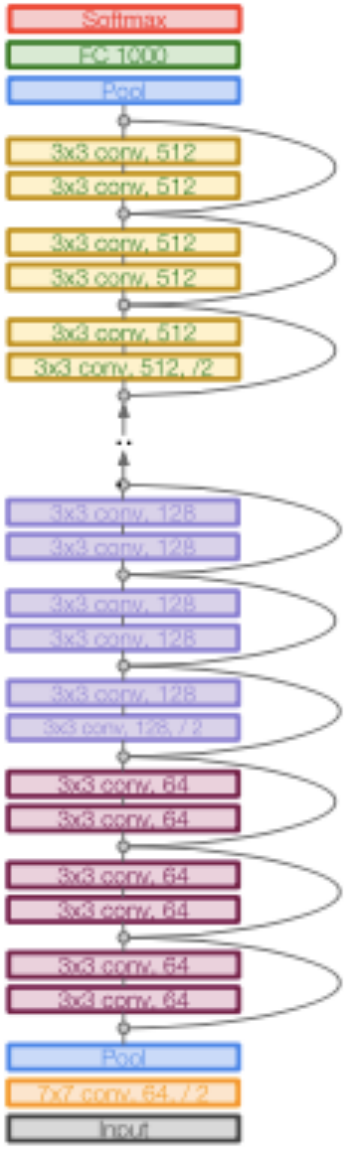
Normal layers

Learn the correction factor

Solves the vanishing gradients problem!

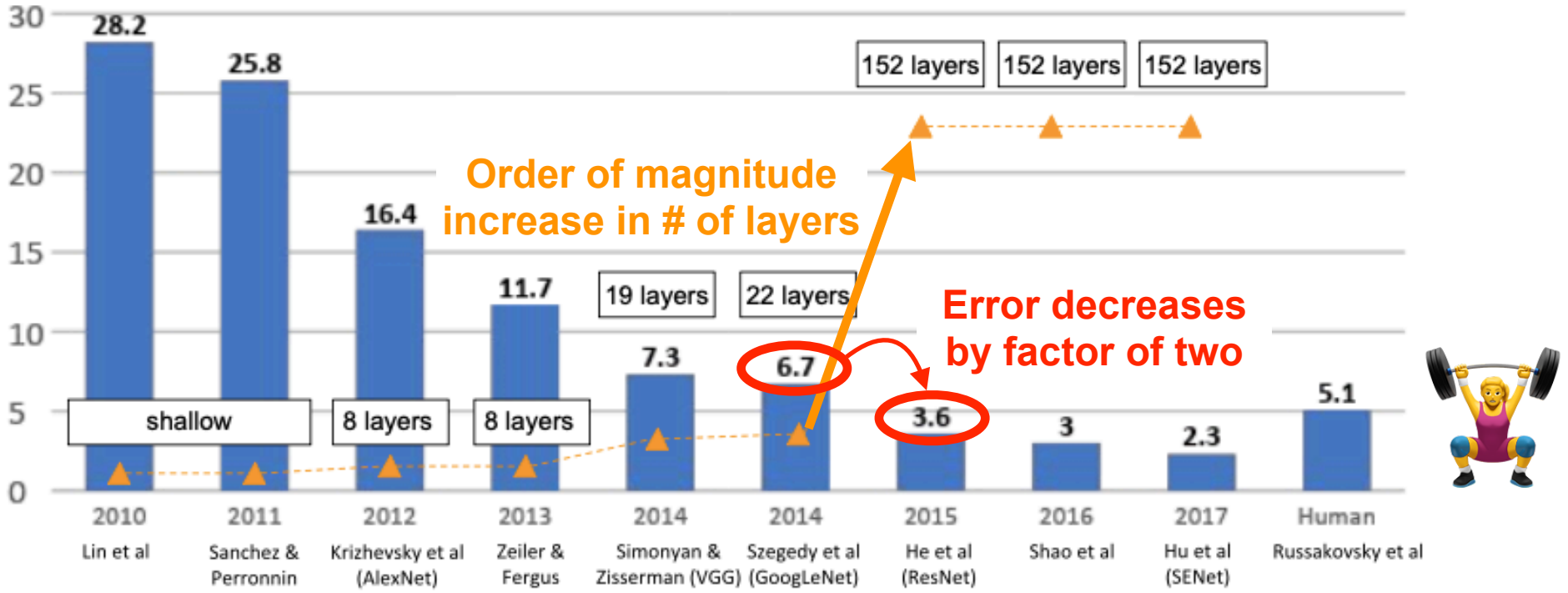
Residual block

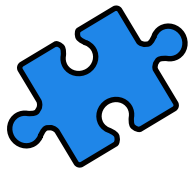
ResNet: performance



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

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

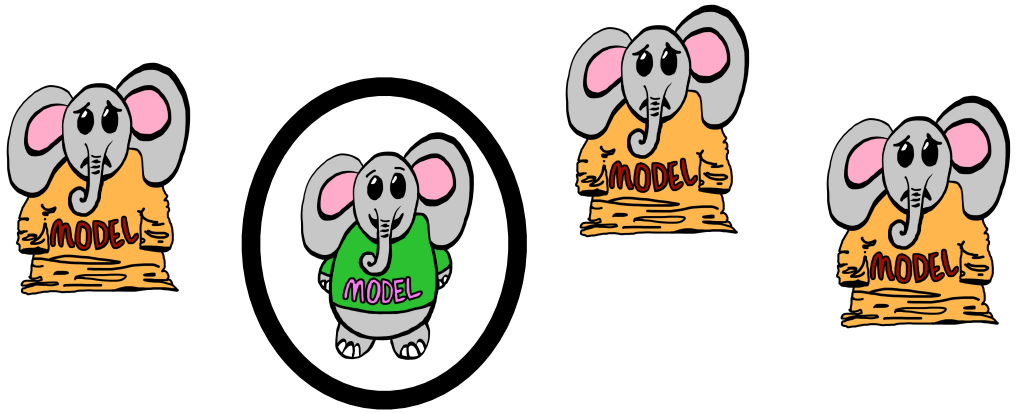
Fit the training data

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!



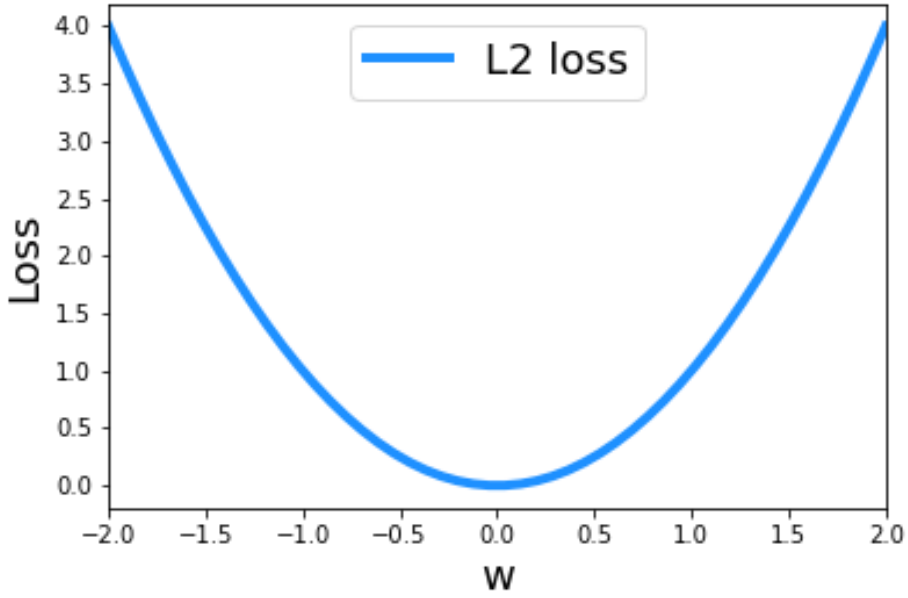
L2 Regularization (most common for NNs)

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

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

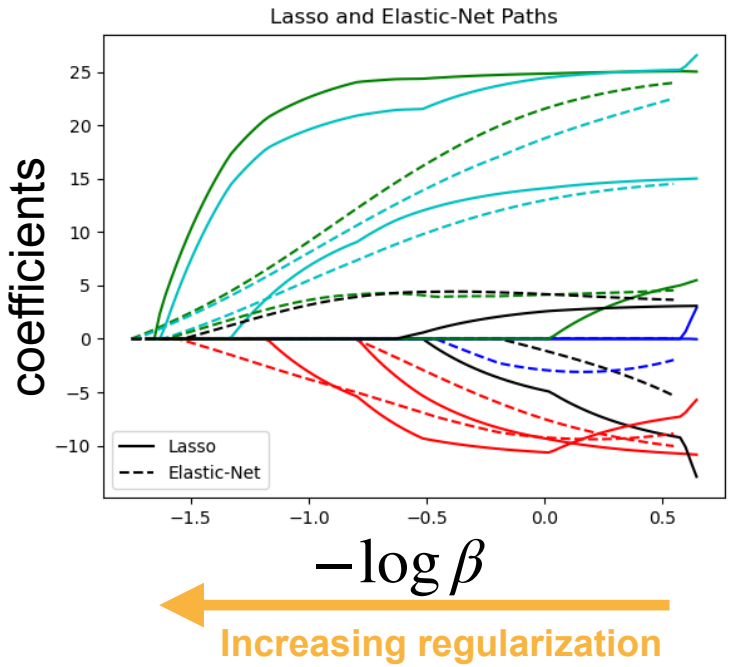
(Log scale, e.g, β = 0, 1e-6, 1e-4)

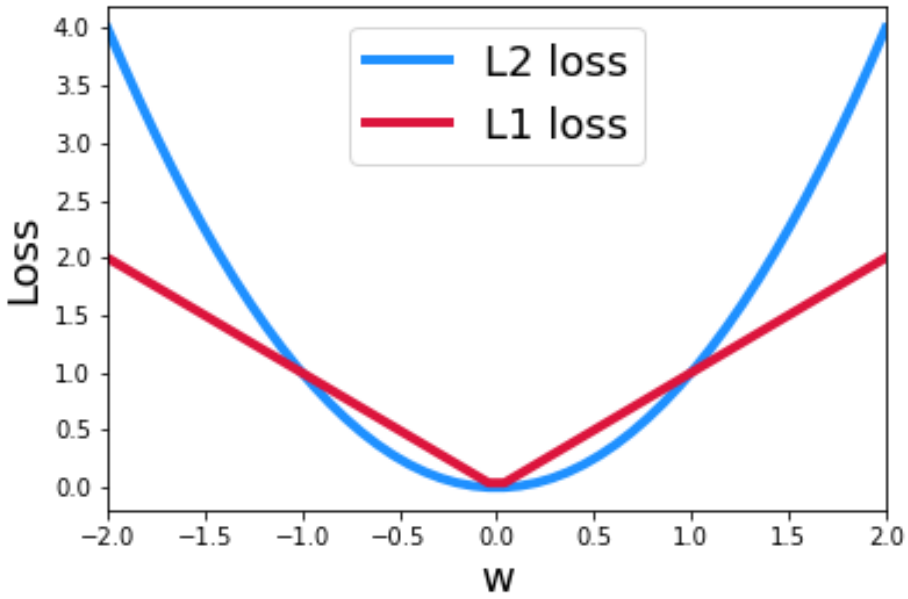
scikit learn
application in 1804.06913

L1 regularization (inducing sparsity)

$$\mathscr{L}_\beta(w) = \mathscr{L}(w) + \beta \, |w|$$

Encourages weights to be *small*



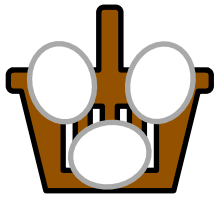


Srivastava et al, “Dropout: A simple way to prevent neural networks from overfitting”, JMLR 2014

CS231n Lecture 7

Dropout: intro

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

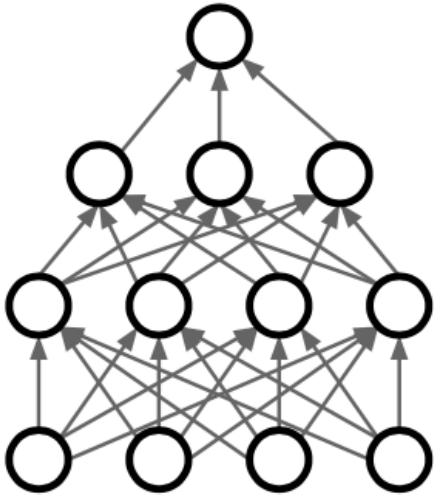


ML version of not putting all your eggs in one basket

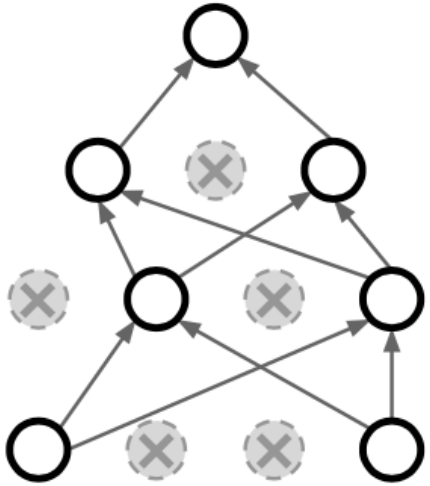
Dropout: idea

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

Original Network



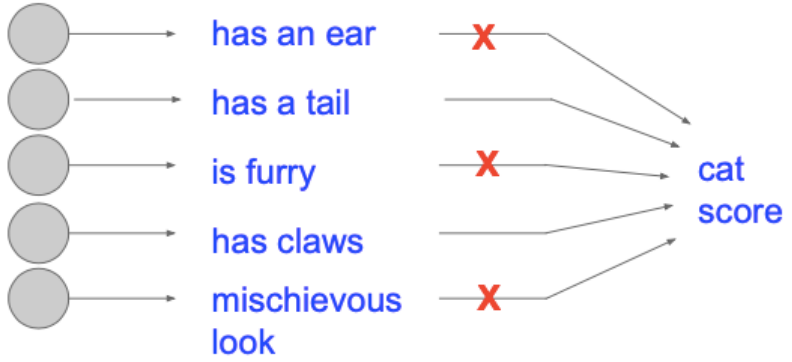
With dropout



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

Hyperparameter we need to optimize!

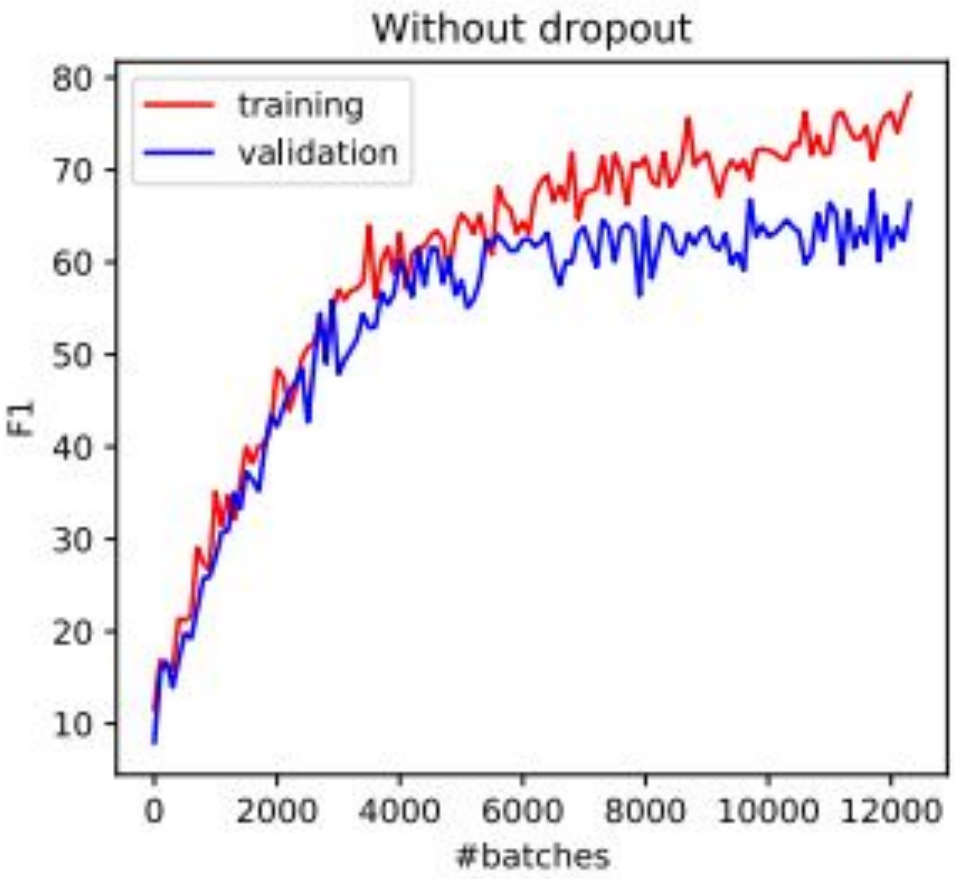
✅ Encourages learning robust features

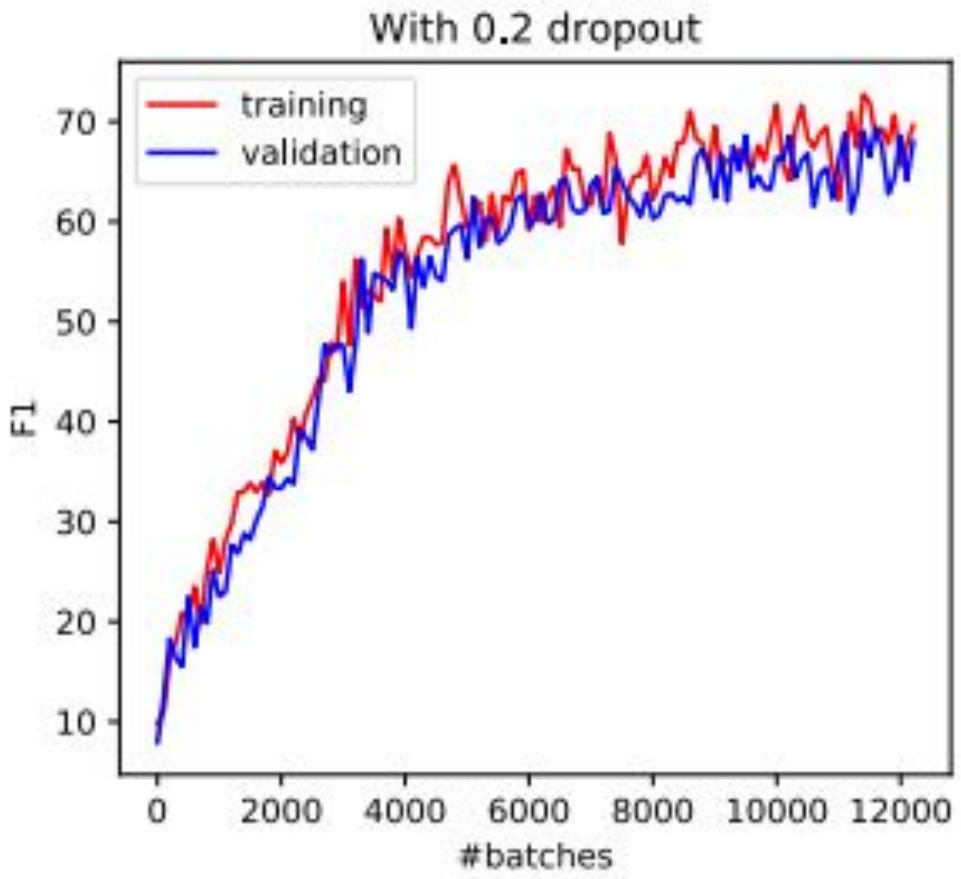


At **test time**, use all the neurons for prediction!!

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

Dropout: performance



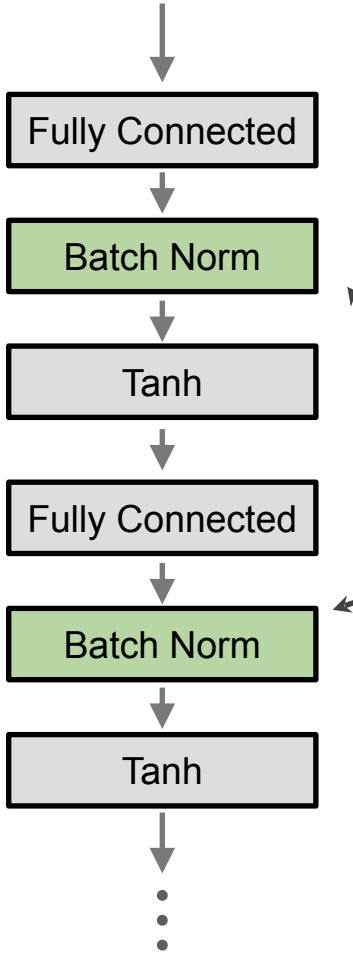


Overfit first — then try to close the gap!!

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

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

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

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

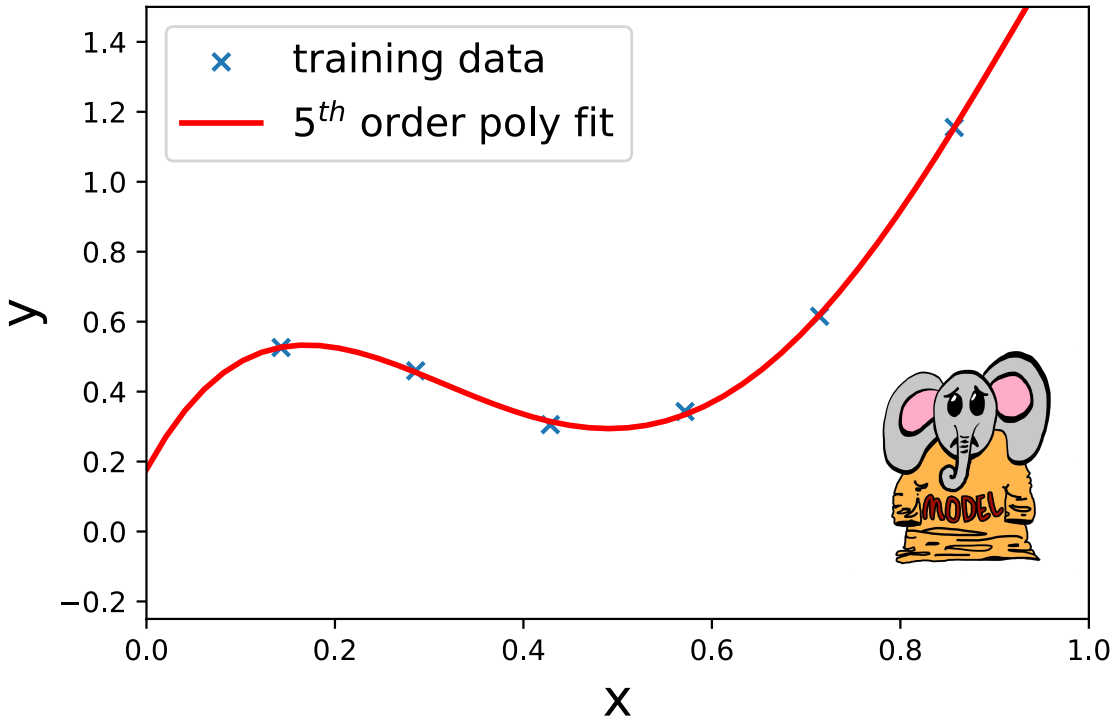
Original paper: inserted after Fully Connected layers, and before nonlinearity.

Some debate, e.g, 1912.04259 suggests other placements better in some cases.

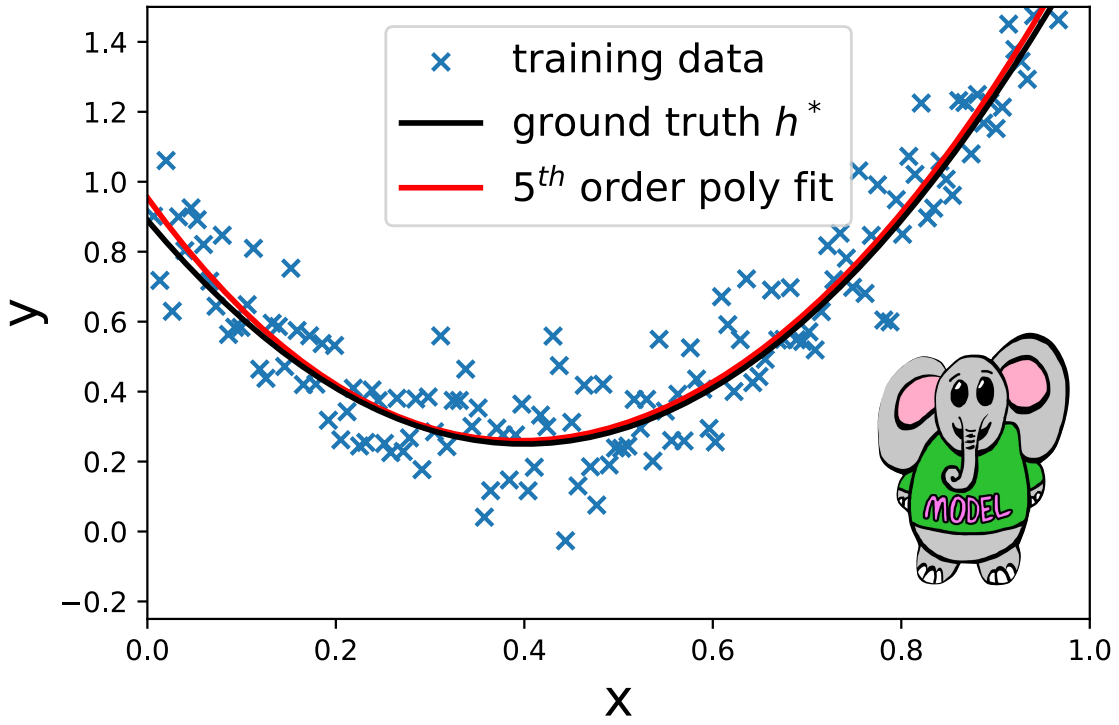
For set / sequence based data (Nikolai's talk) LayerNorm also useful

Data augmentation : motivation

Recall: More training data reduces variance.



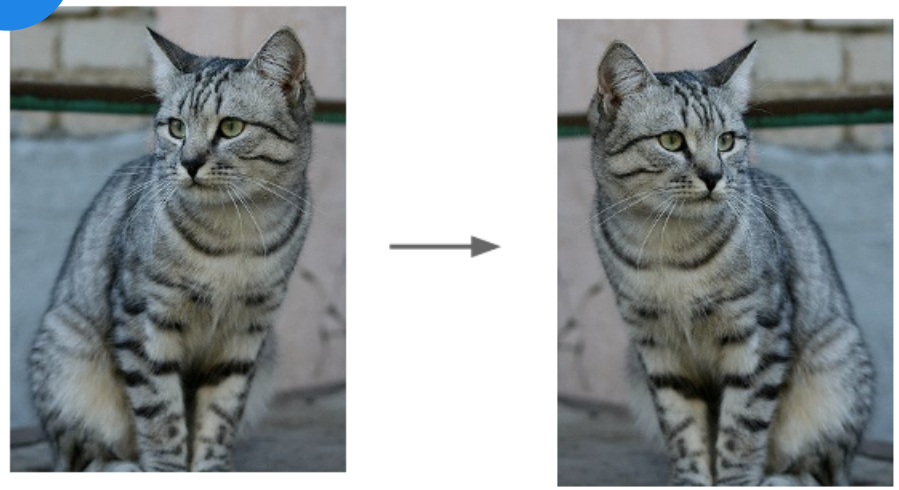
Fit 5^{th} order polynomial on large dataset



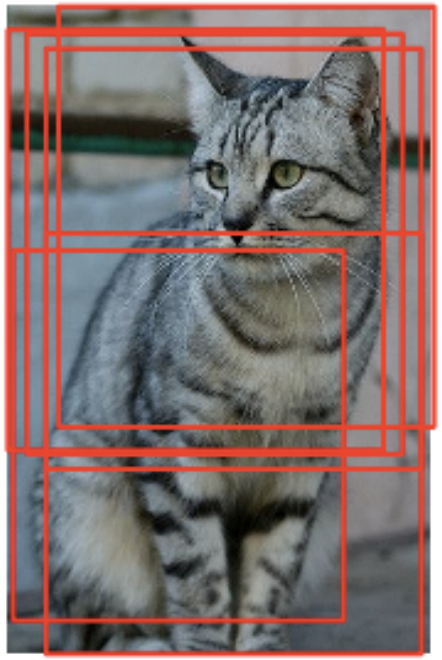
Q: How can you modify your training data to artificially increase your dataset size?

Data augmentation : examples

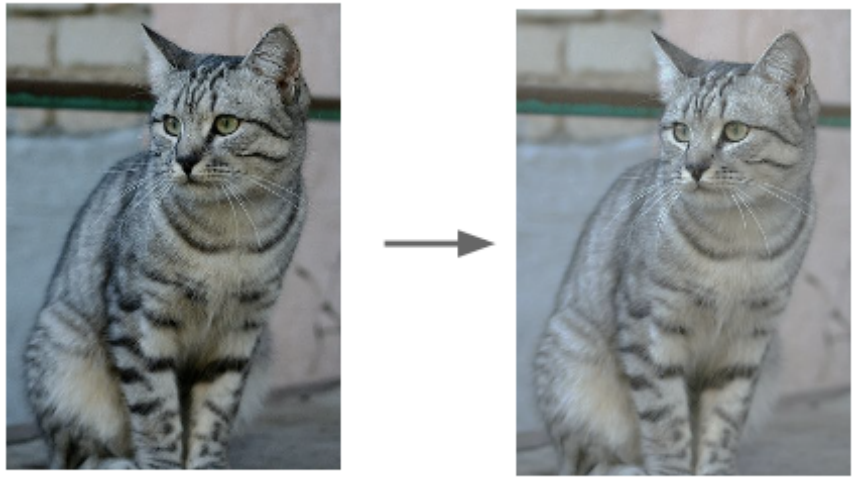
1 Horizontal flips



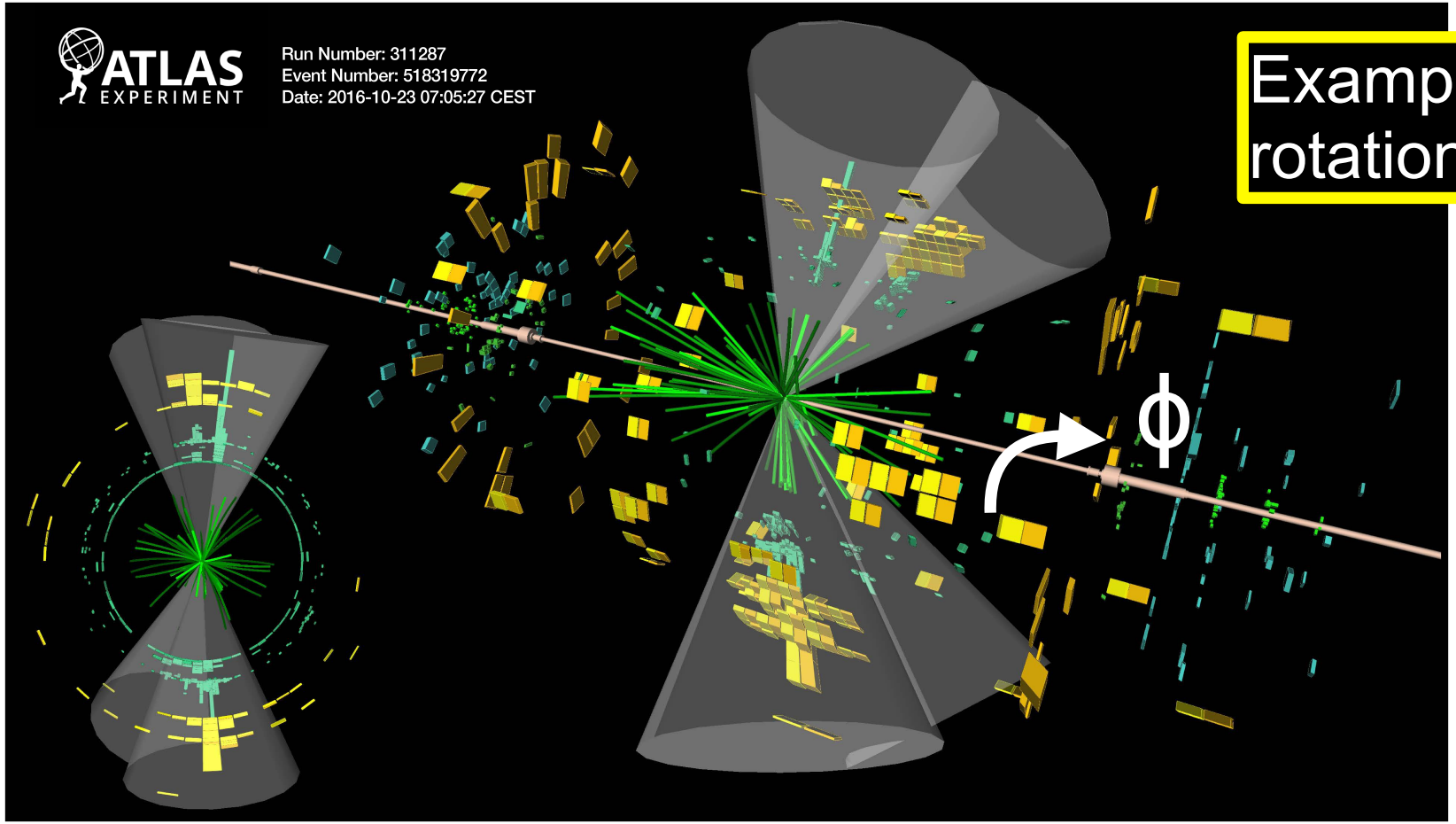
2 Random crops and scales



3 Adjust contrast and brightness



Data augmentation — physics



Less used in practice…

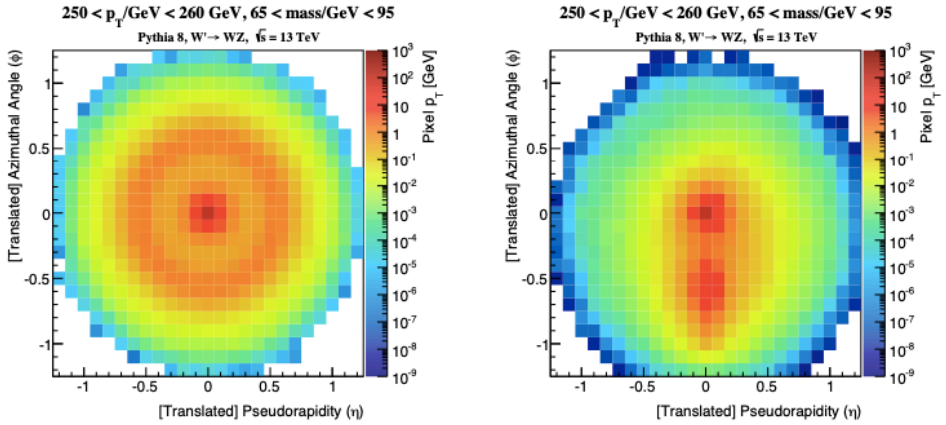
With a simulator and we can often get as many training examples as we want

Data augmentation — alternatives

Issue: Larger models with more data take longer to train

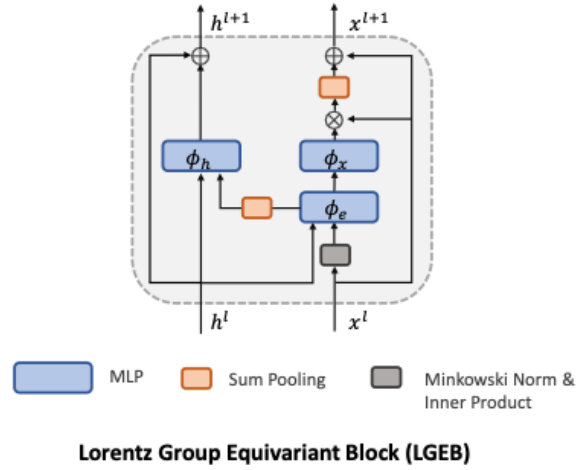
➡ Remove the variation to train faster.

1 Preprocess for uniformity



Jet Images — Deep Learning Edition 1511.05190

2 Architecture design to preserve invariants

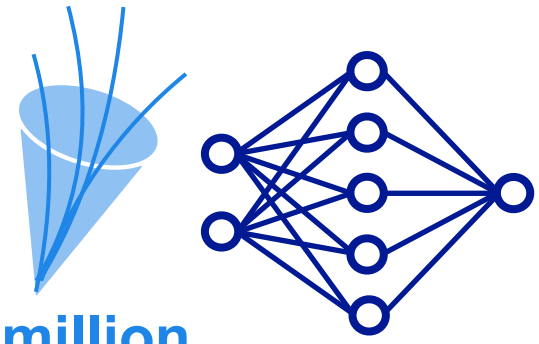


Lorentz Group Equivariant Block (LGEB)

LorentzInvariance: Lorentz Net and ParT 2202.03772
Azimuthal Symmetry: 2107.02908
+ many others

Fine-tuning / transfer learning

Original Task

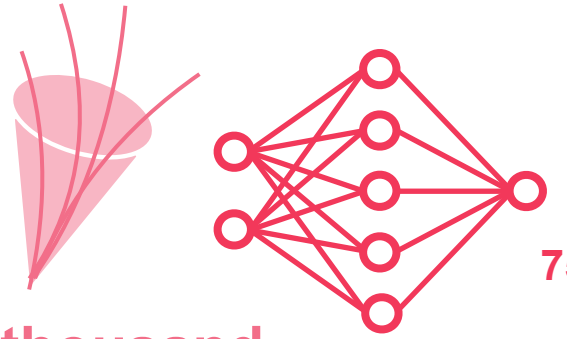


x 20 million

Particle ID

95% accurate

Modified task



x 200 thousand

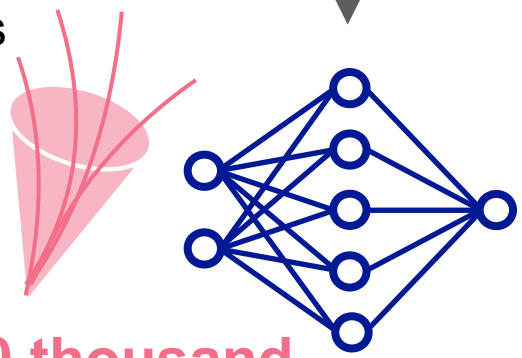
Particle ID

75% accurate

Fine-tuning

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

(2) Adjust weights with the smaller dataset.



x 200 thousand

Particle ID

95% accurate

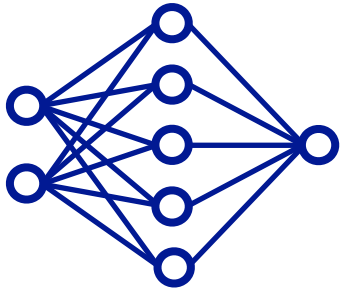
(3) Enjoy increase in performance!

Fine-tuning / Transfer learning: chatGPT

Original Task

GPT-3

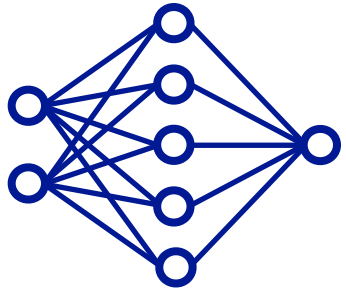
Train on the corpus
of the whole internet



Fine-tuning

chatGPT

Fine-tune weights
on the chatbot
converations



O I want you to act as a Julia REPL. I will type commands and you will reply with what the REPL should show. I want you to only reply with the REPL output inside one unique code block, and nothing else. Do not write explanations. Do not type commands unless I instruct you to do so. When I need to tell you something in English I will do so by putting text inside curly brackets {like this}. My first command is VERSION.

Copy code

```
julia> VERSION
v"1.7.0"
```

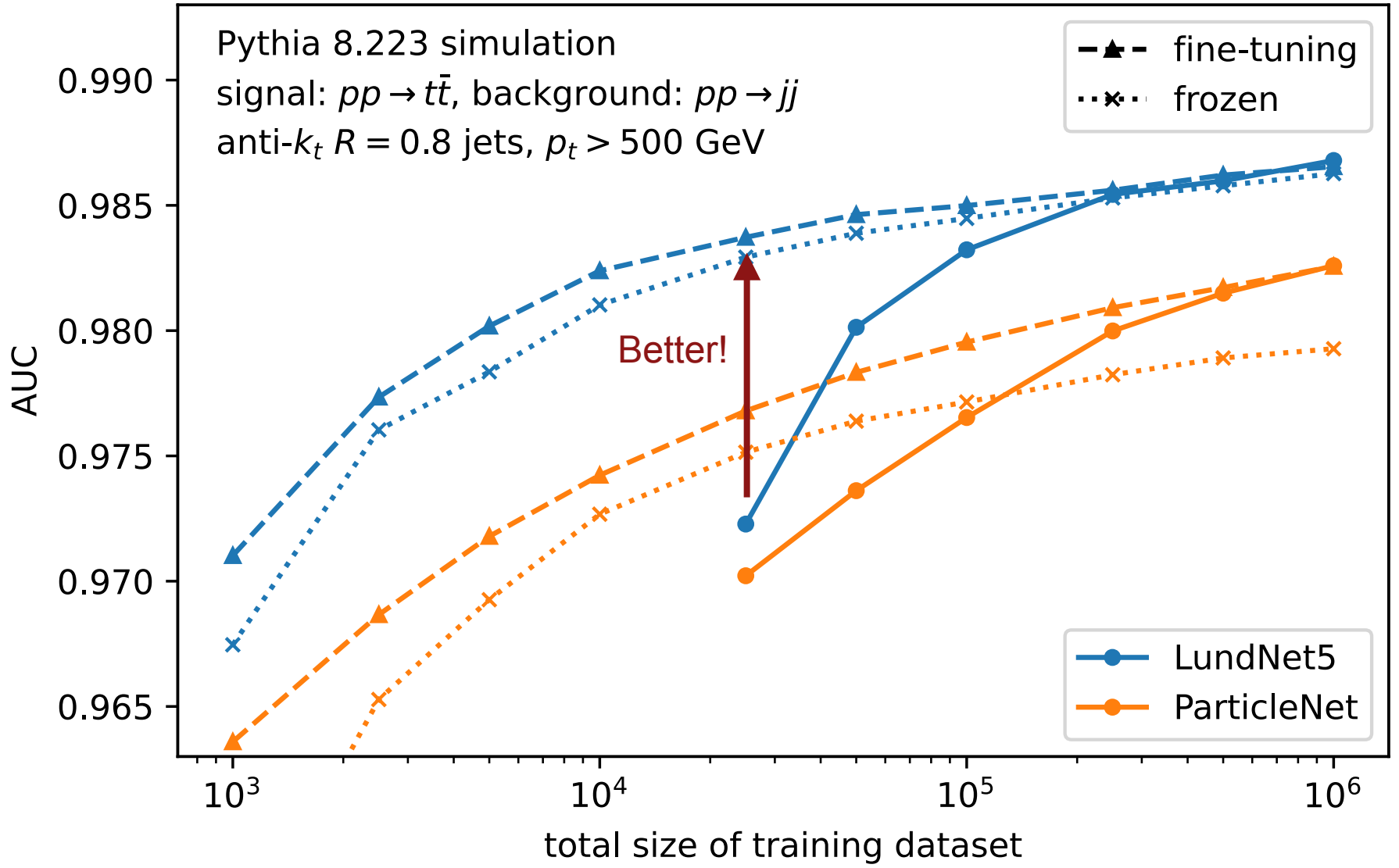
O cholesky([2.0 0.5; 0.5 1.2])

Copy code

```
julia> cholesky([2.0 0.5; 0.5 1.2])
LowerTriangular{Float64,Array{Float64,2}}
  [1.141  0.0  ]
  [0.447  0.785]
```

Apparently… chatGPT has learned math and julia syntax

Fine-tuning / Transfer learning

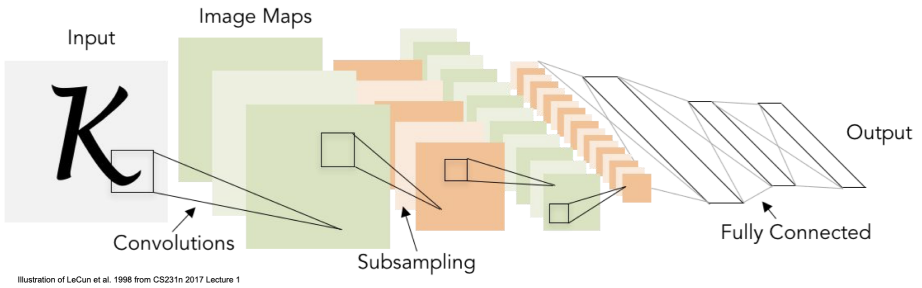


Weight sharing

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

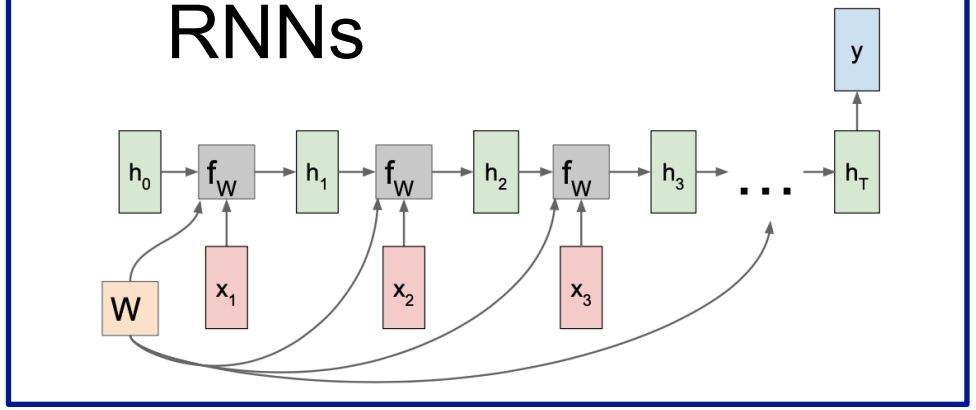
"Don't relearn what you don't need"

CNNs



See next talk by Judith!

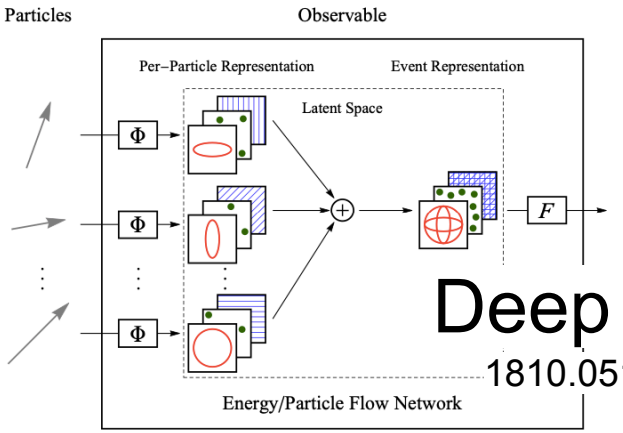
RNNs



See tomorrow by Nikolai!

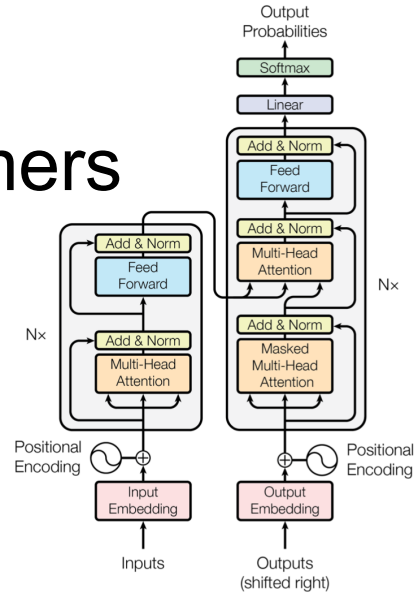
Deep Sets

1810.05165



Transformers

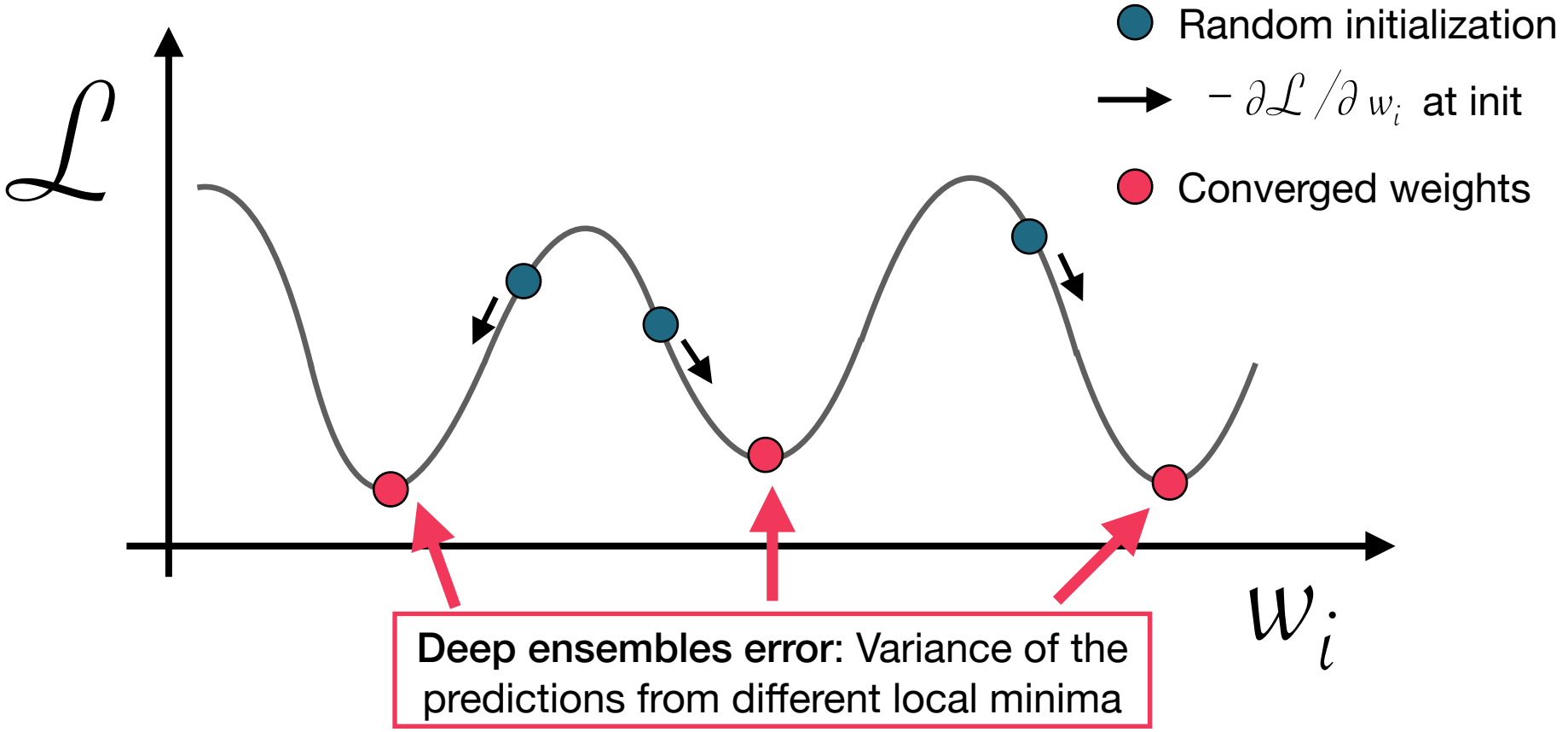
1706.03762



Ensembles

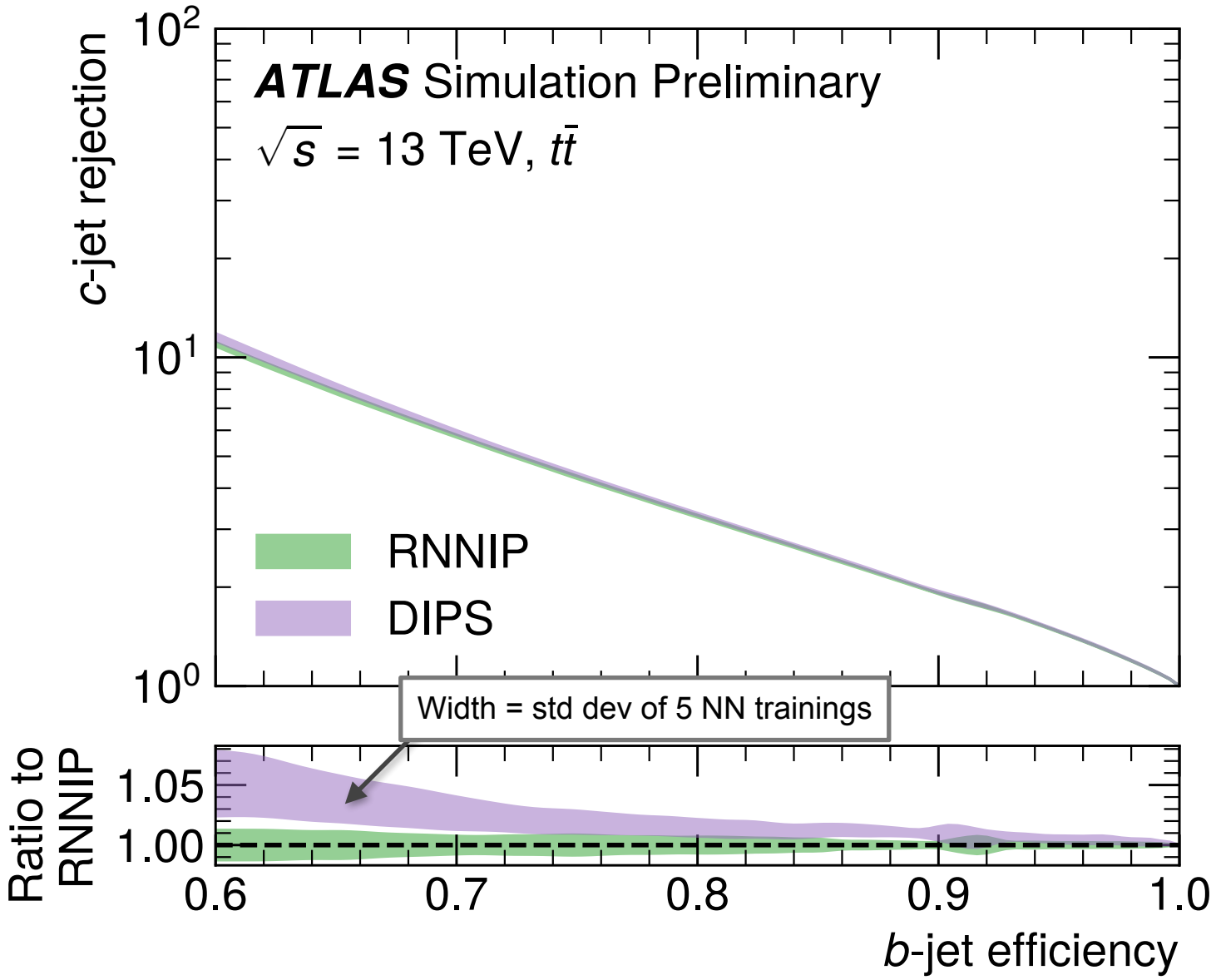
$$MSE(x) \approx (h^*(x) - h_{avg}(x))^2 + \mathbb{E}\left[(h_{avg}(x) - h_S(x))^2\right]$$

Variance

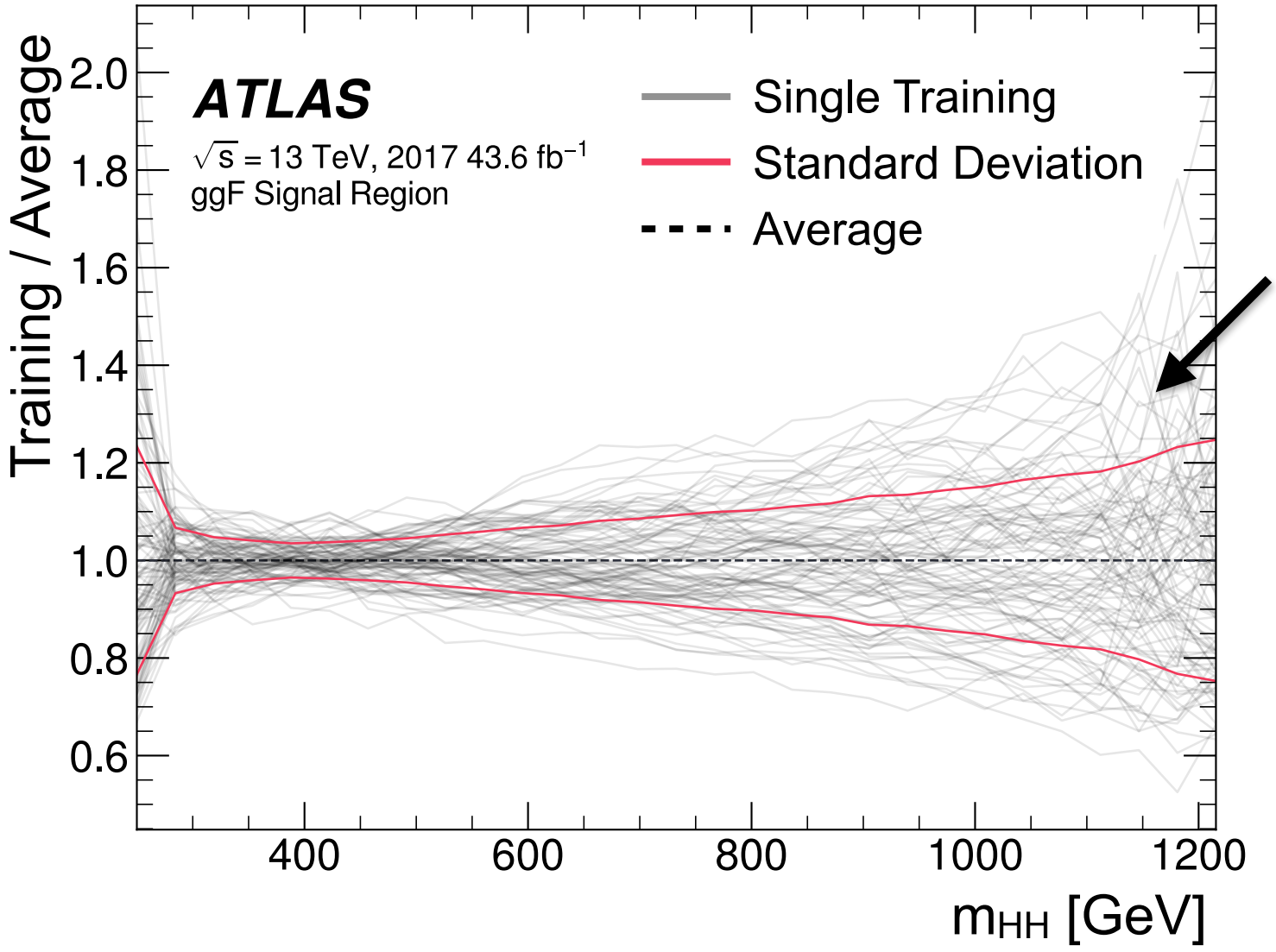


Ensembles: Application

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

Feature choices

Training techniques

Statistical learning theory

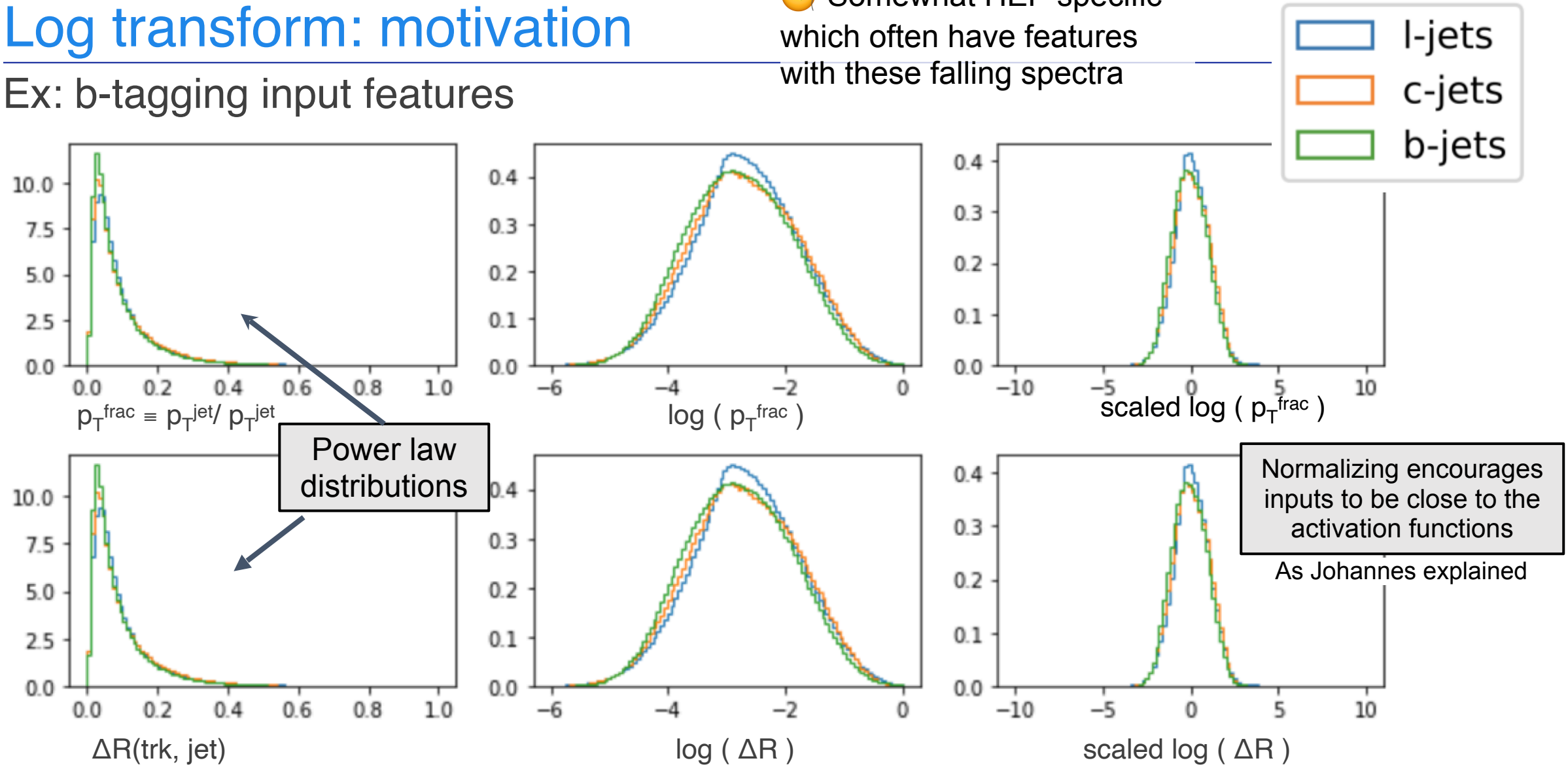
Bias / variance trade-off

Would love feedback + discussions!

Log transform: motivation

Ex: b-tagging input features

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

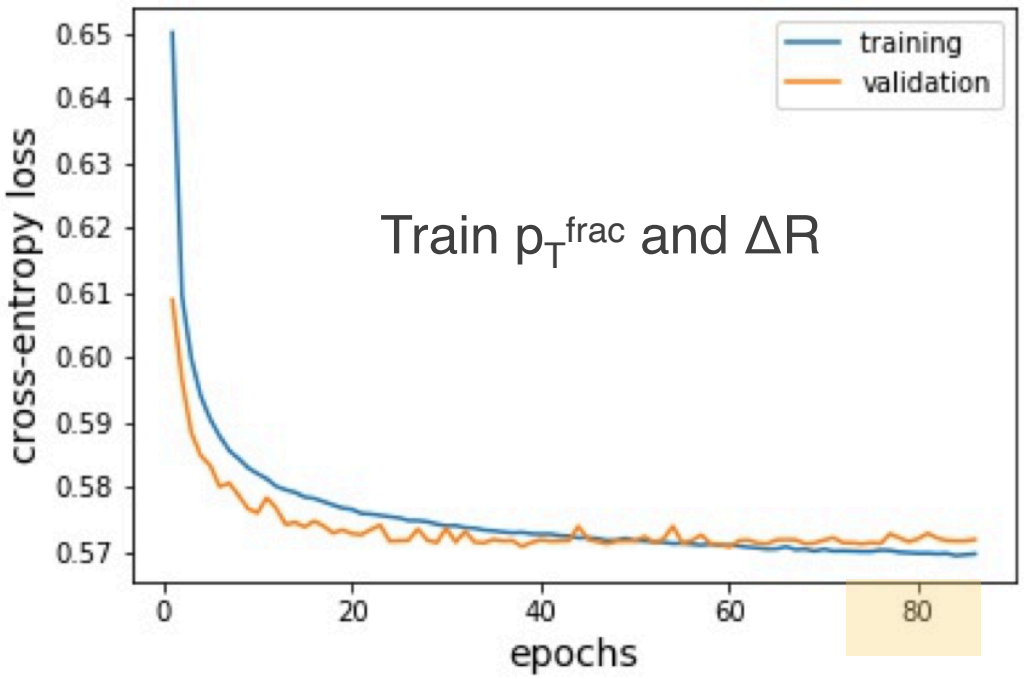


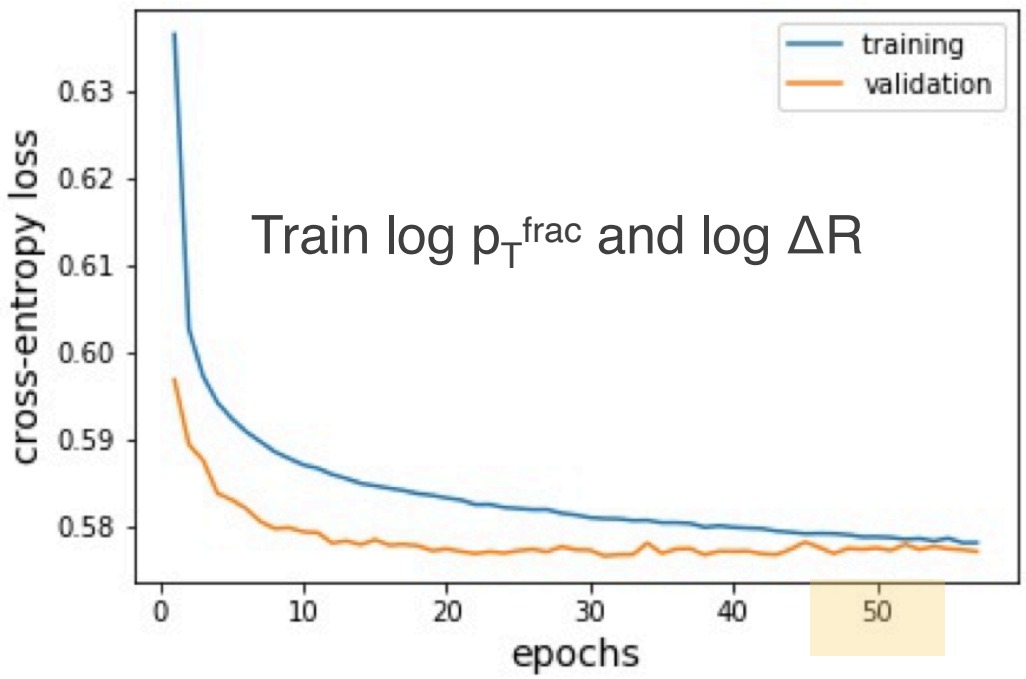
Normalizing encourages inputs to be close to the activation functions

As Johannes explained

With the log, become bell-shape

Log transform: motivation

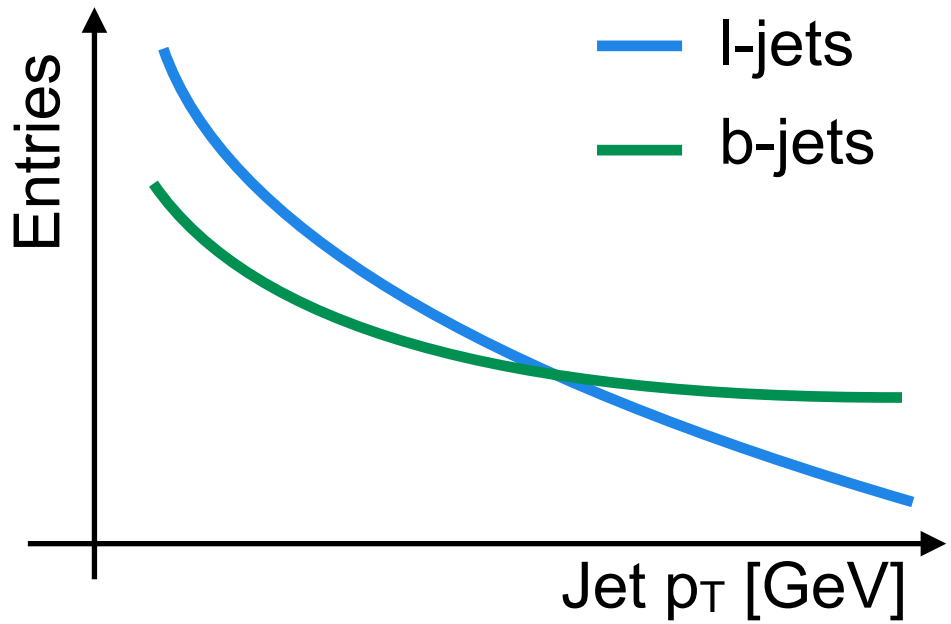


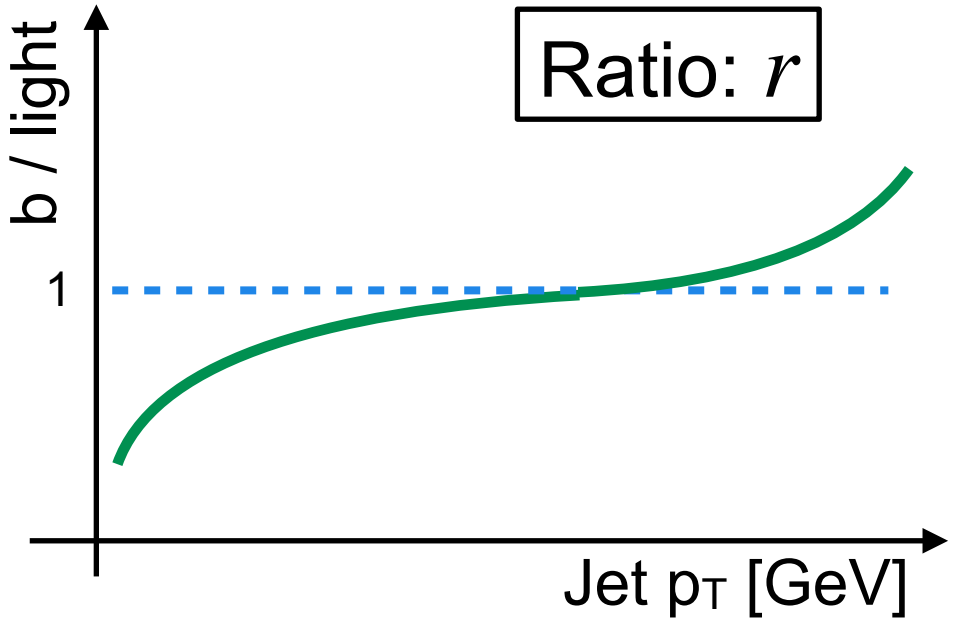


How does this help? 20% speed up in training time!

Sample dependence

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





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

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

Can also use **class weights** for imbalanced classes.

Ablation studies: What has the model learned?

Example — spam classification

RC **Random Company**
Useless Ad
⋮

❌ **Spam**

TK **Thomas Kuhr**
TTT workshop
⋮

✅ **Not Spam**

What made the difference?

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]

Saliency maps

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

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

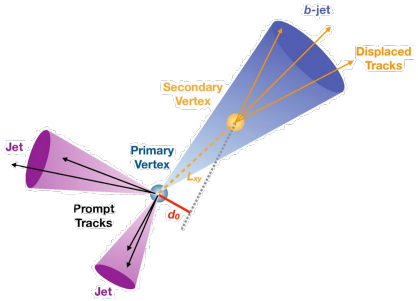
$$\theta = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

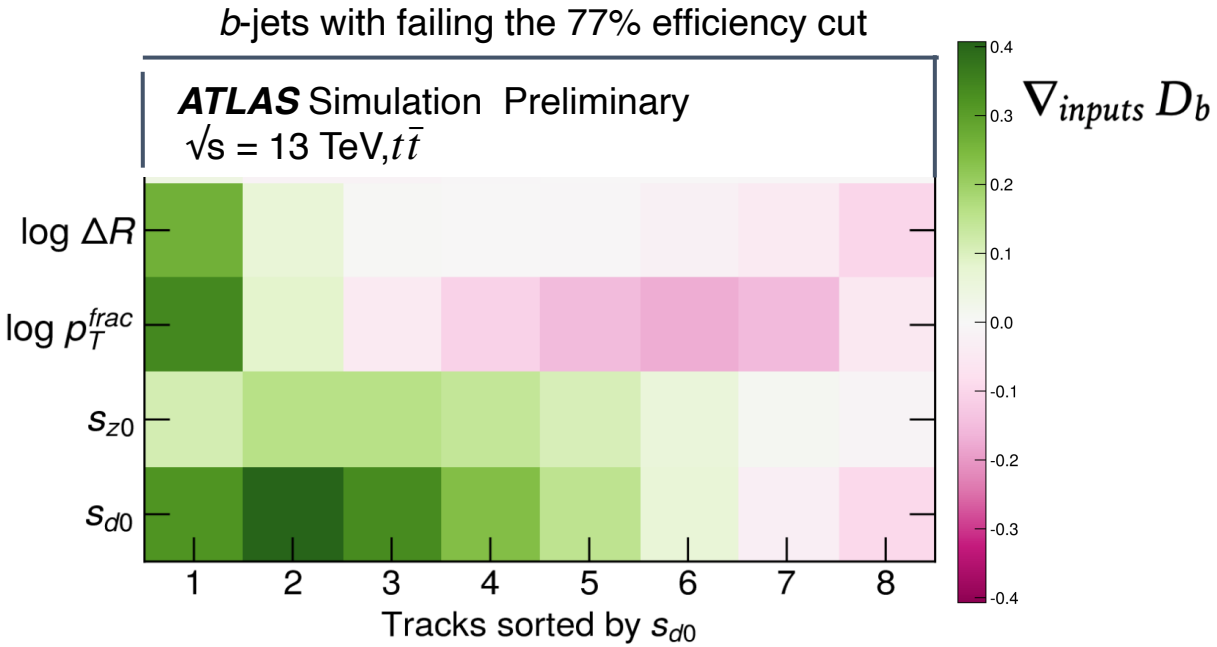
Saliency map: Plot the $|\theta|$ for each of these inputs

Saliency maps

Physics

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

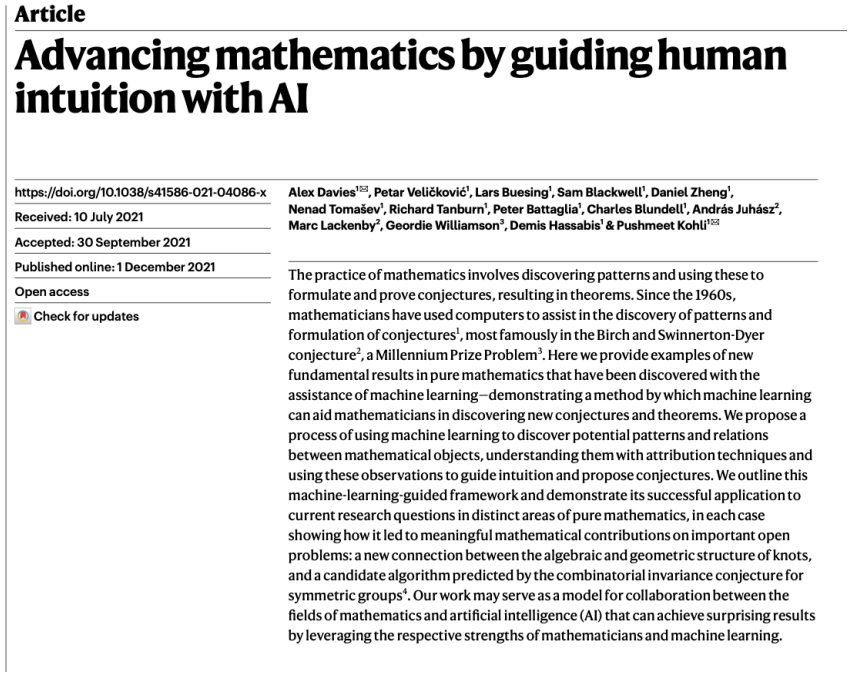




ATL-PHYS-PUB-2020-014

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[1,2], 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[3], Geordie Williamson[3], Demis Hassabis[1] & Pushmeet Kohli[1,2]

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)

Debugging exercise

[If ⏰ permits]

Loss curves — what are the problems?

Hypotheses

Slow start: initialization learning rate too small

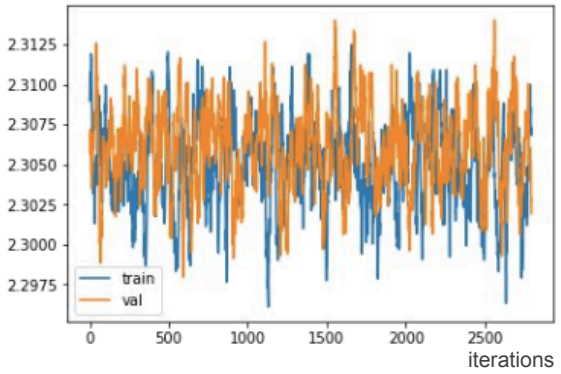
Applied the negative of the gradients

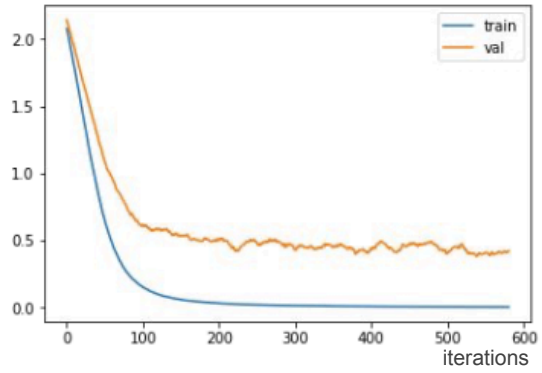
Not converged yet: need longer training

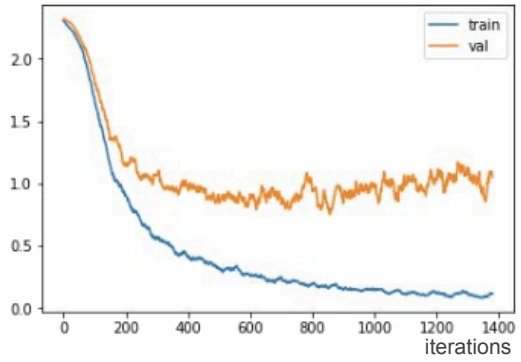
Not learning: gradients not applied to the weights

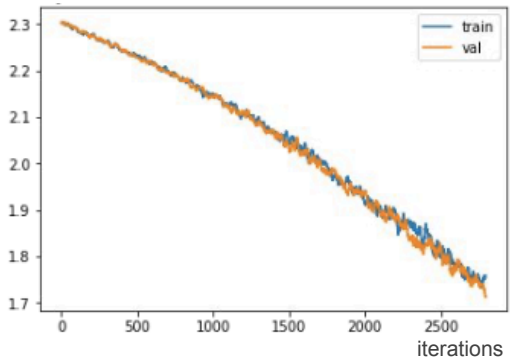
Overfit: model too large / dataset too small

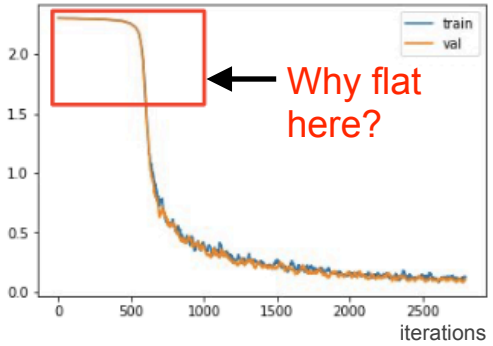
Loss curves

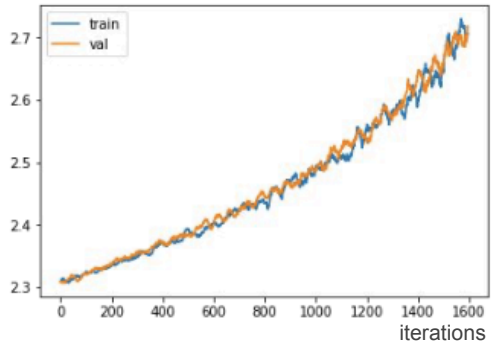












Starting off…

Open question

Feature choices

Going deeper

Training techniques

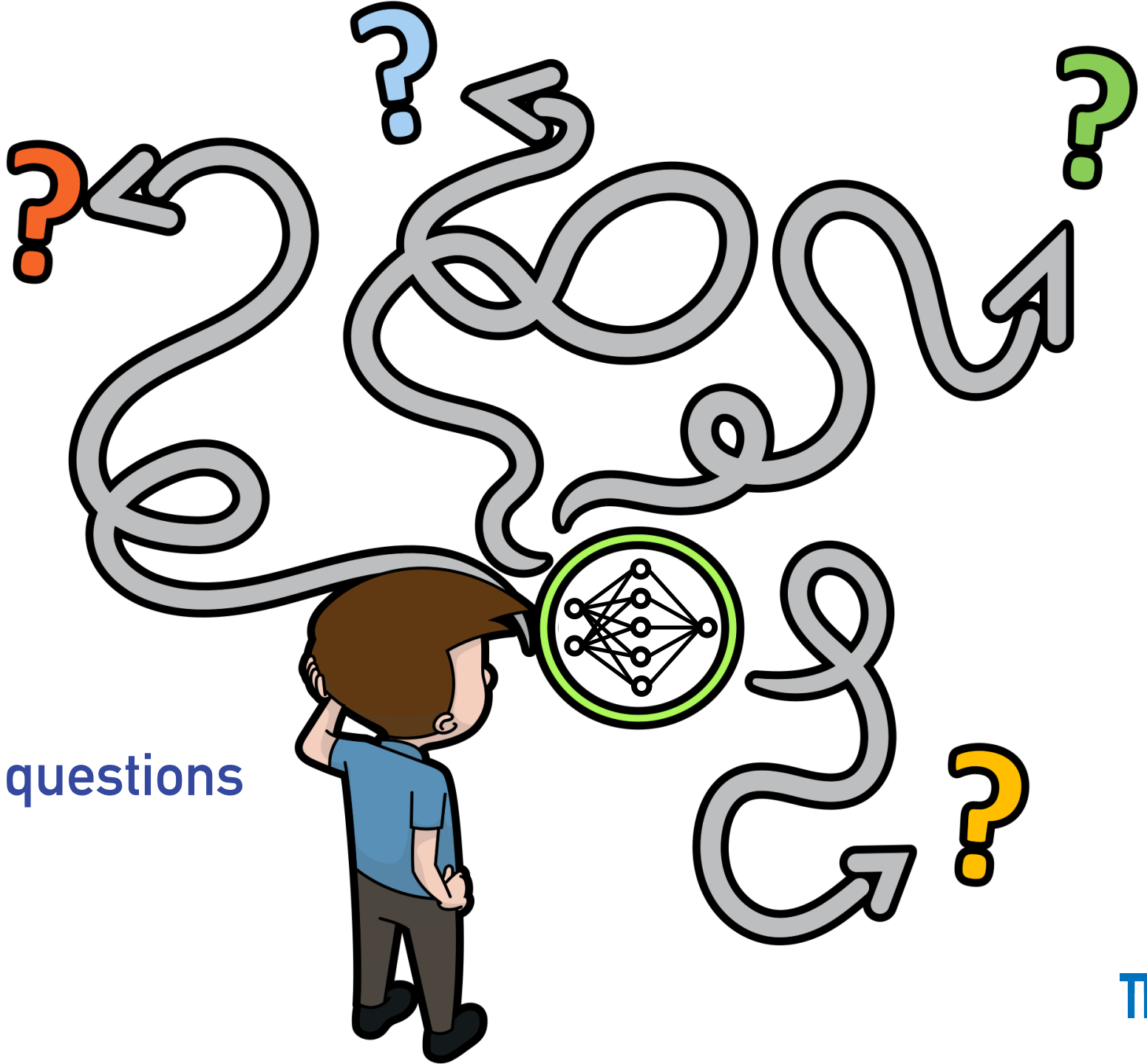
Statistical learning theory

Bias / variance trade-off

✋ Would love feedback + discussions!

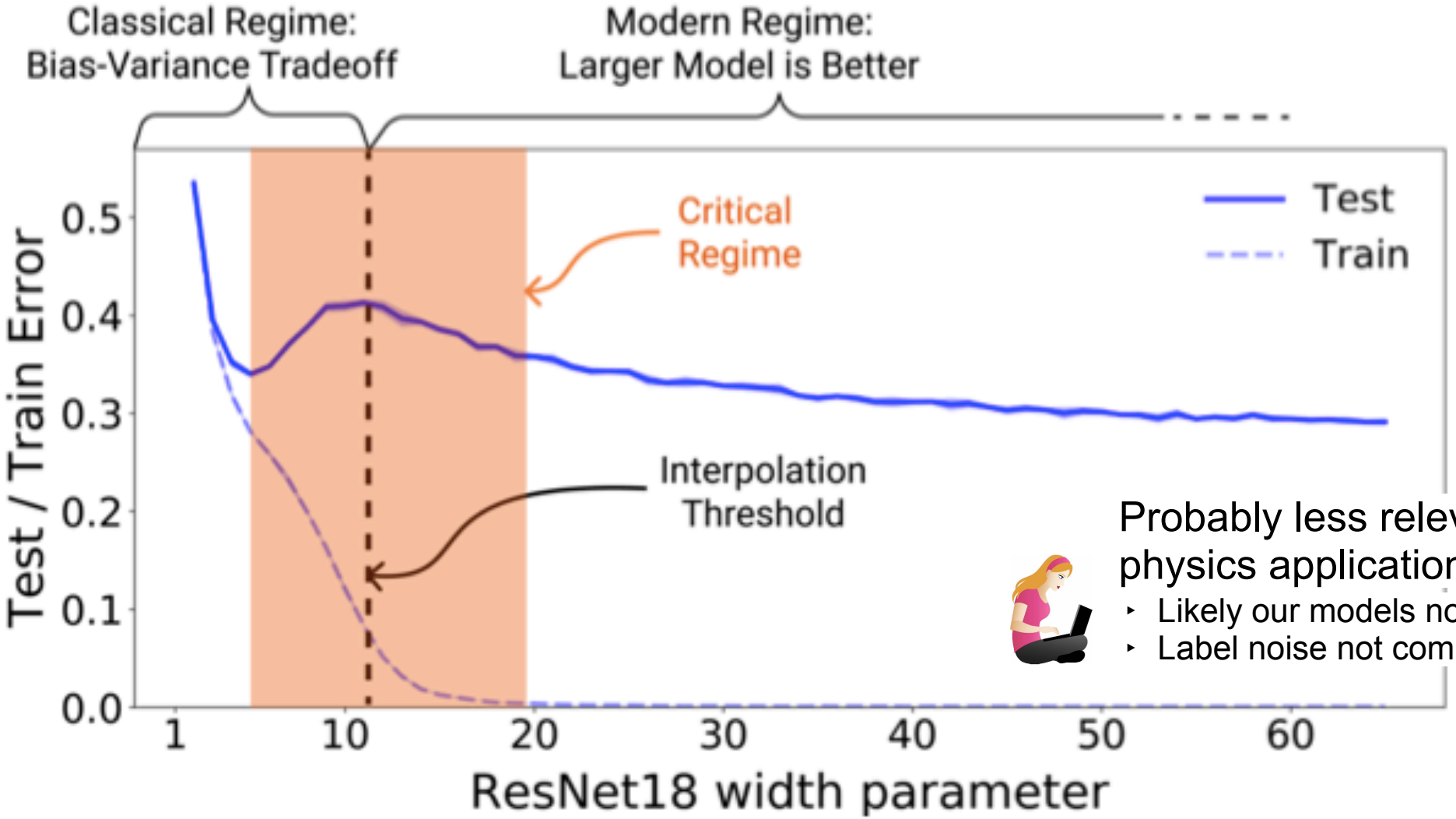
[One of the]

Outstanding deep questions in deep learning.



Double Descent

1912.02292
Daniela Witten
@daniela_witten
post



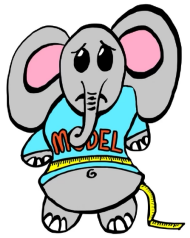
Probably less relevant for physics applications

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

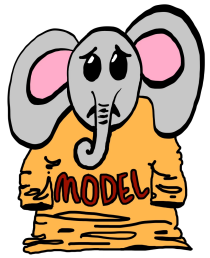
What did we learn today about choosing the right model?

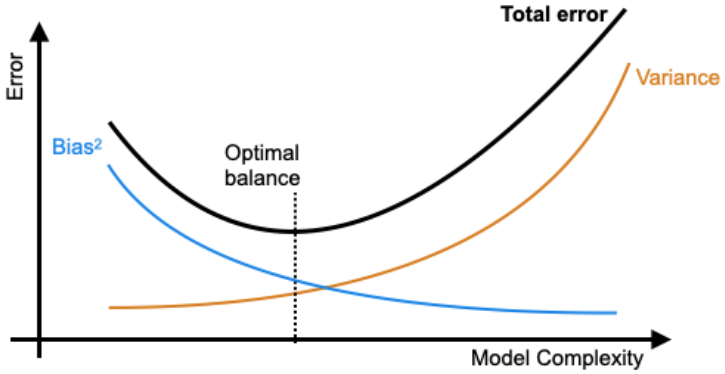
Total error = Bias^2 + Variance

Underfitting

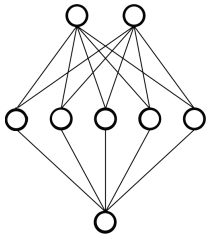


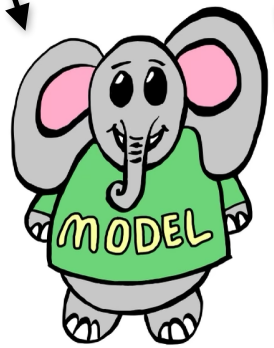
Overfitting





Deep learning gains: increasing complexity





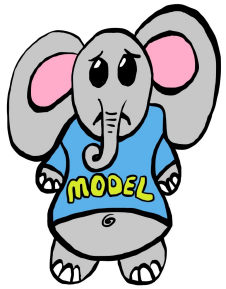
Regularization:

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

Training 101:

- ☑ Monitor loss curves for learning rate
- ☑ Use random search for hyper-parameters
- ☑ Use ensembles
  - Model performance
  - Error

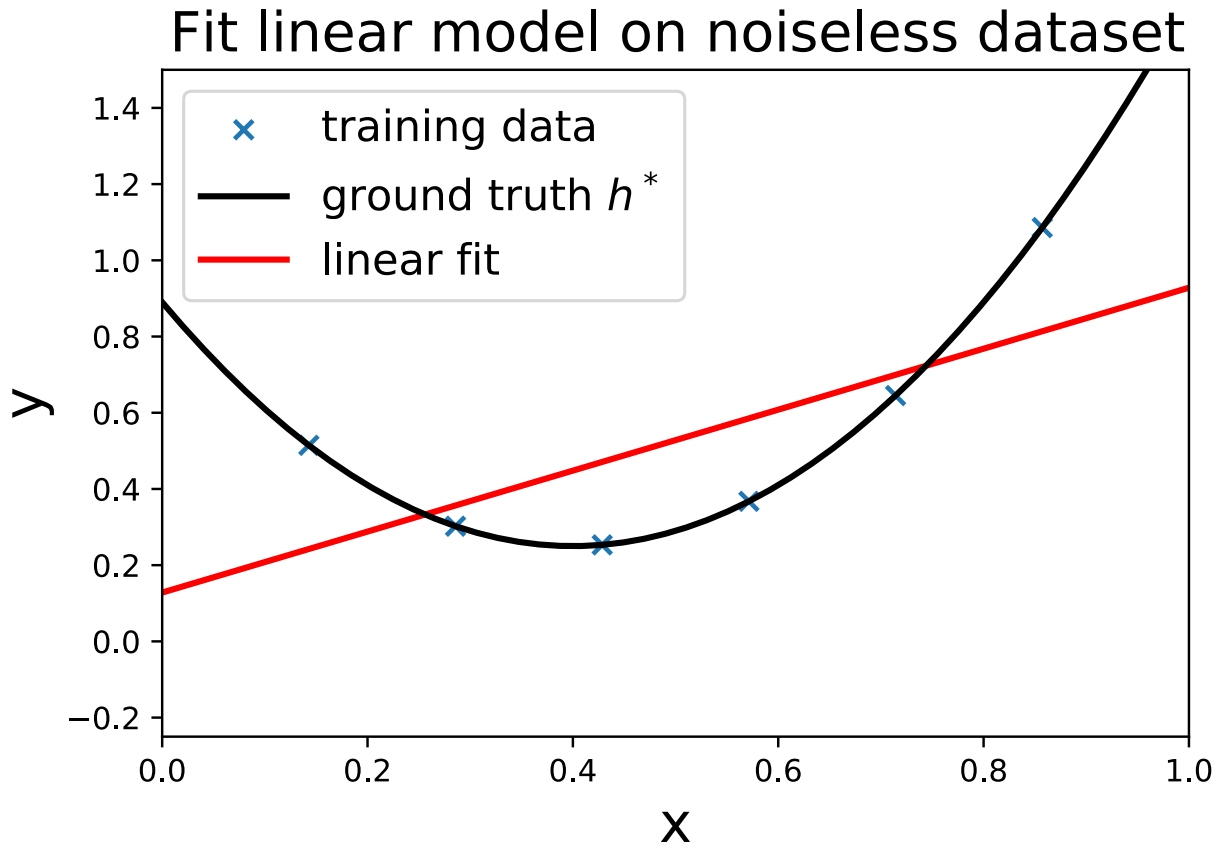
Backup



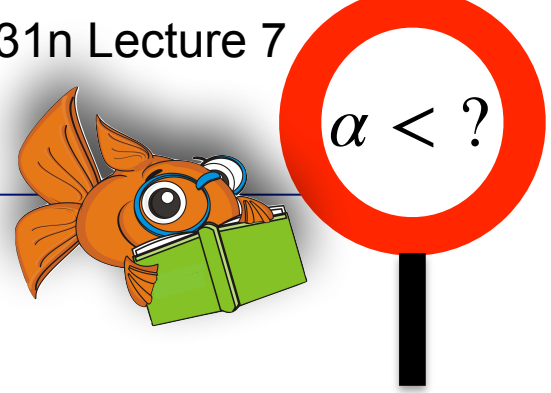
High bias: diagnostics

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

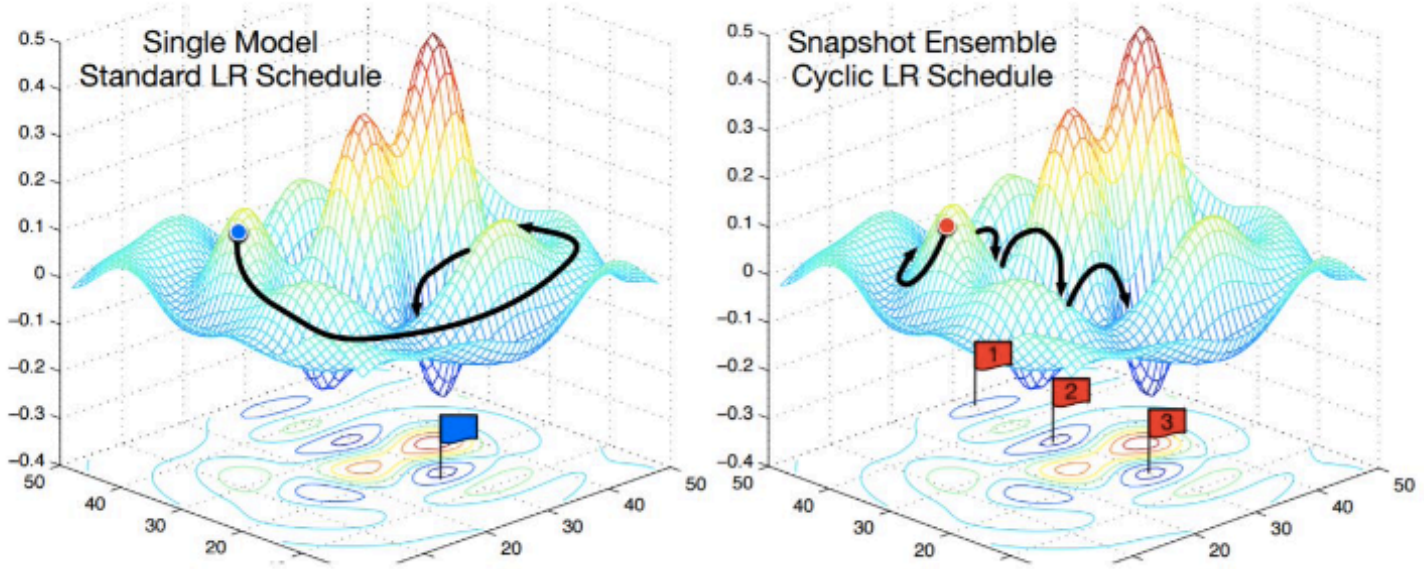
The training error on the linear model still large, even when there is *no noise* on the training data.

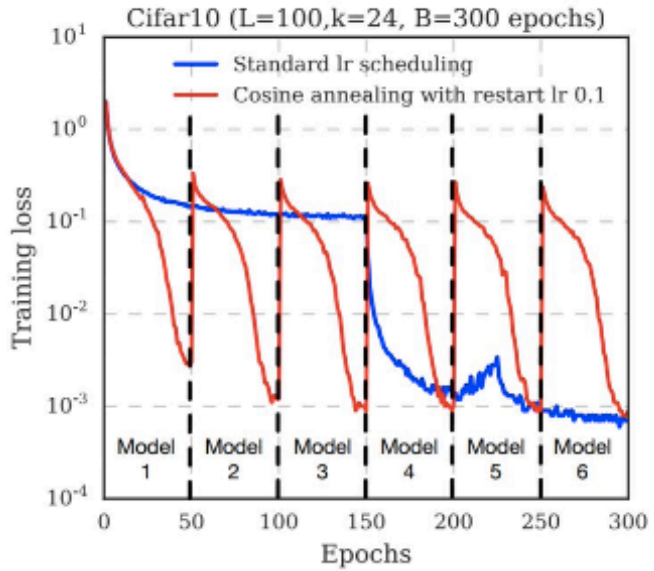


Learning rate schedule



Instead of training independent models, use multiple snapshots of a single model during training!



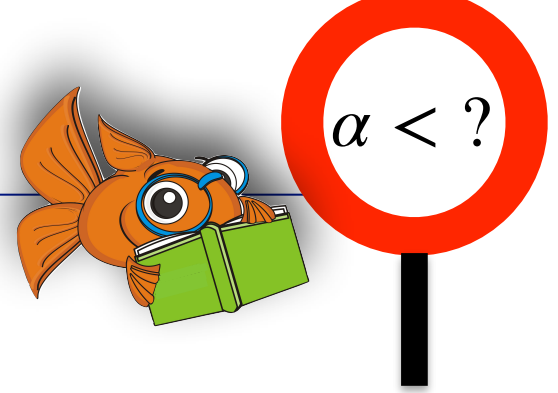


Loshchilov and Hutter, "SGDR: Stochastic gradient descent with restarts", arXiv 2016
Huang et al, "Snapshot ensembles: train 1, get M for free", ICLR 2017
Figures copyright Yixuan Li and Geoff Pleiss, 2017. Reproduced with permission.

Cyclic learning rate schedules can make this work even better!

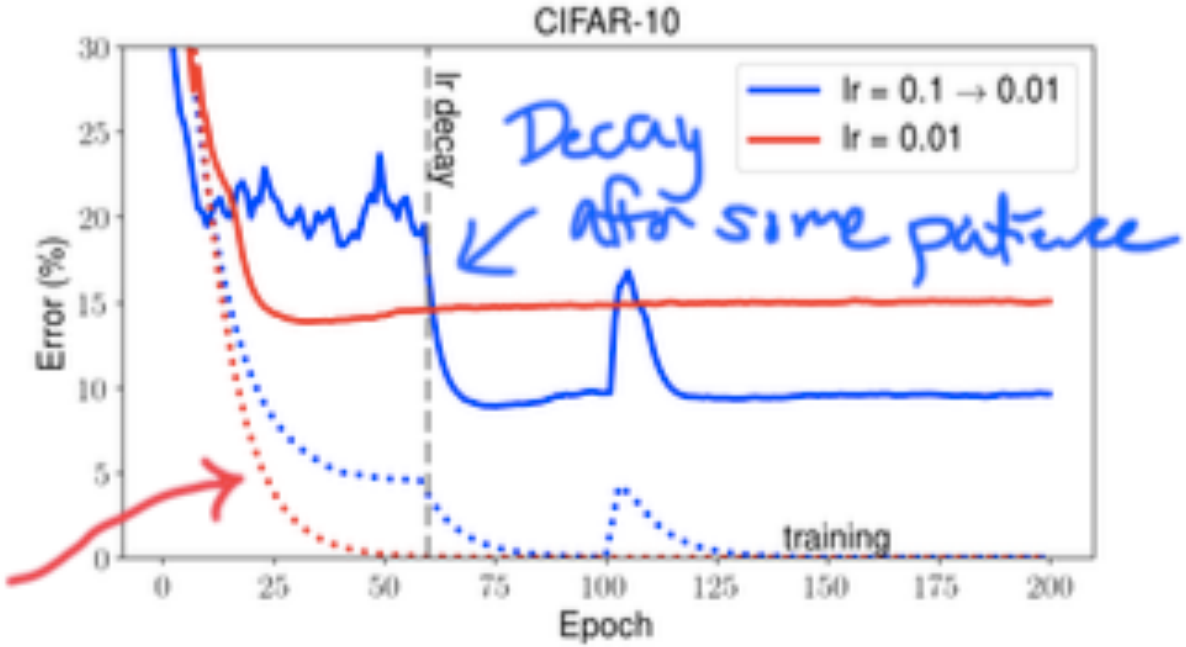
learning rate schedules

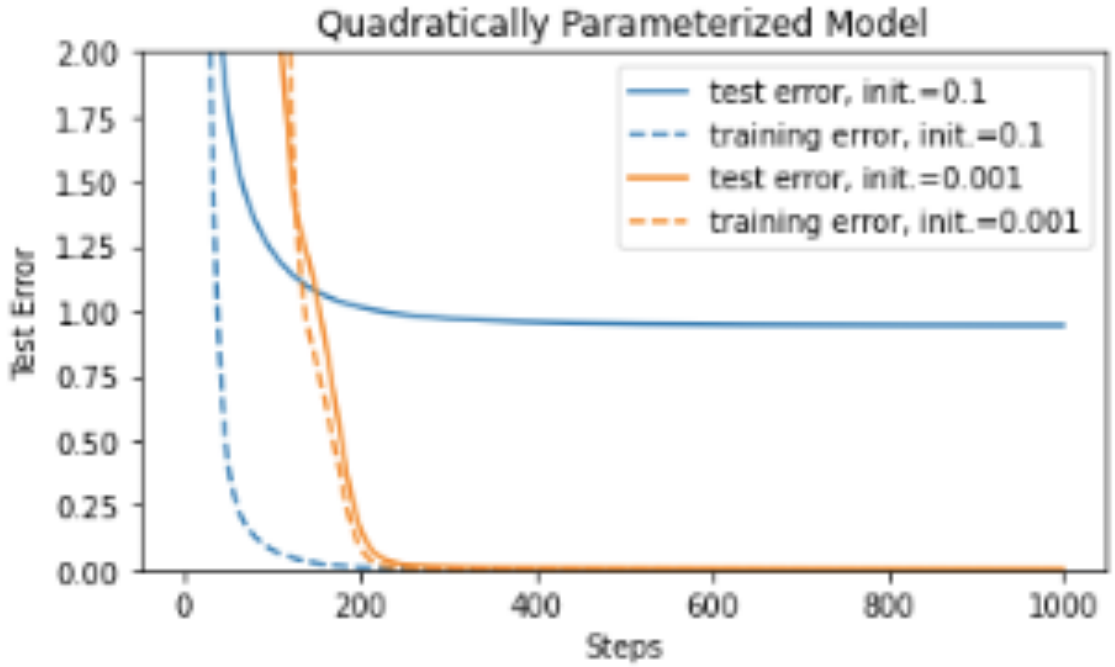
CS229 notes
2002.09277
2006.08680



Minimize $\mathscr{L}$ by SGD

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





Batch size — caveat

Minimize $\mathscr{L}$ by SGD

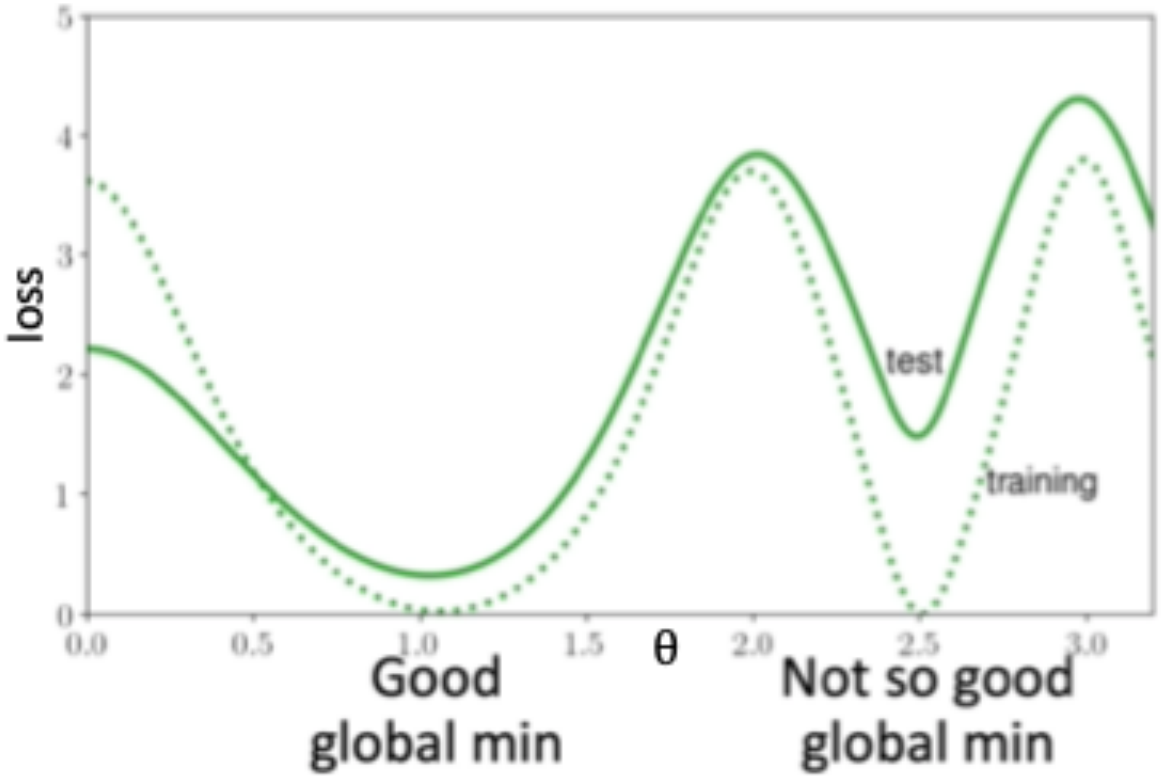
$$w = w - \alpha \boxed{\nabla_w \mathscr{L}} \approx w - \alpha \sum_{i=1}^{m} \nabla_w \mathscr{L}(x_i, y_i)$$

Make a MC estimate

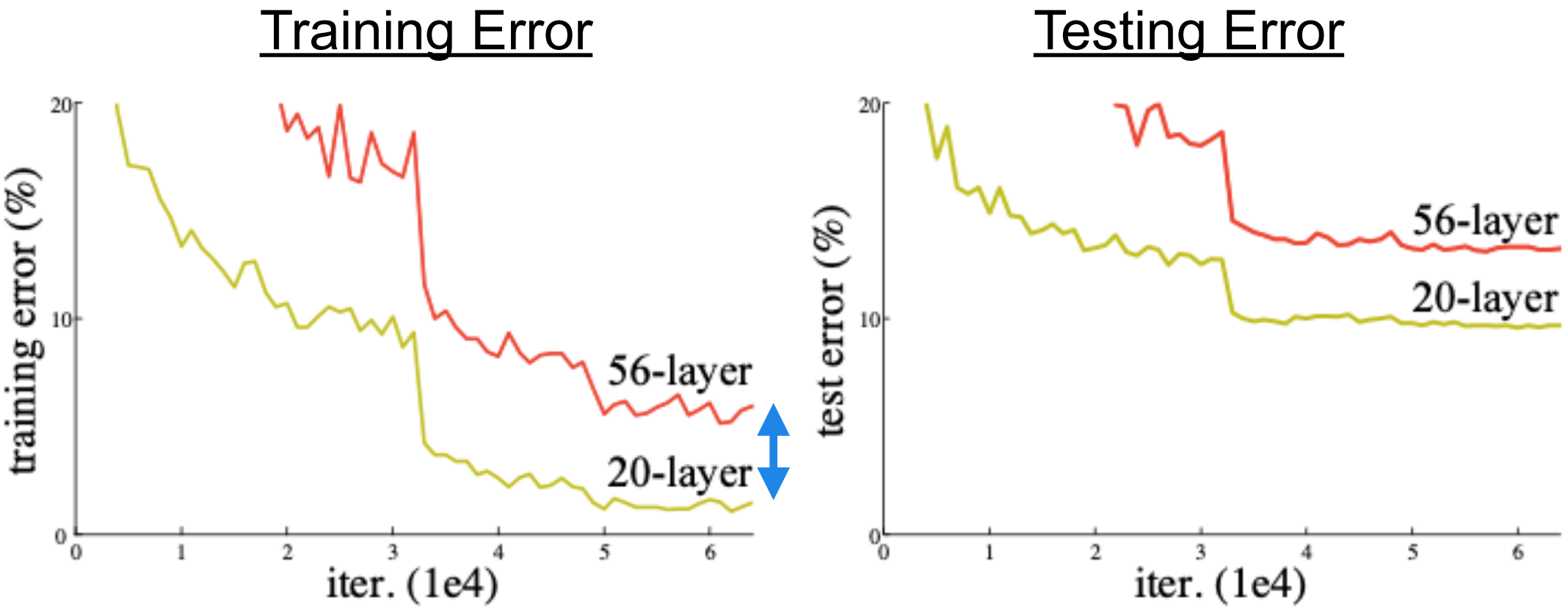
m: mini-batch size

Alternative philosophy…

Smaller mini-batches encourage stochasticity and converge to flatter minima which generalize better.



ResNets: Motivation

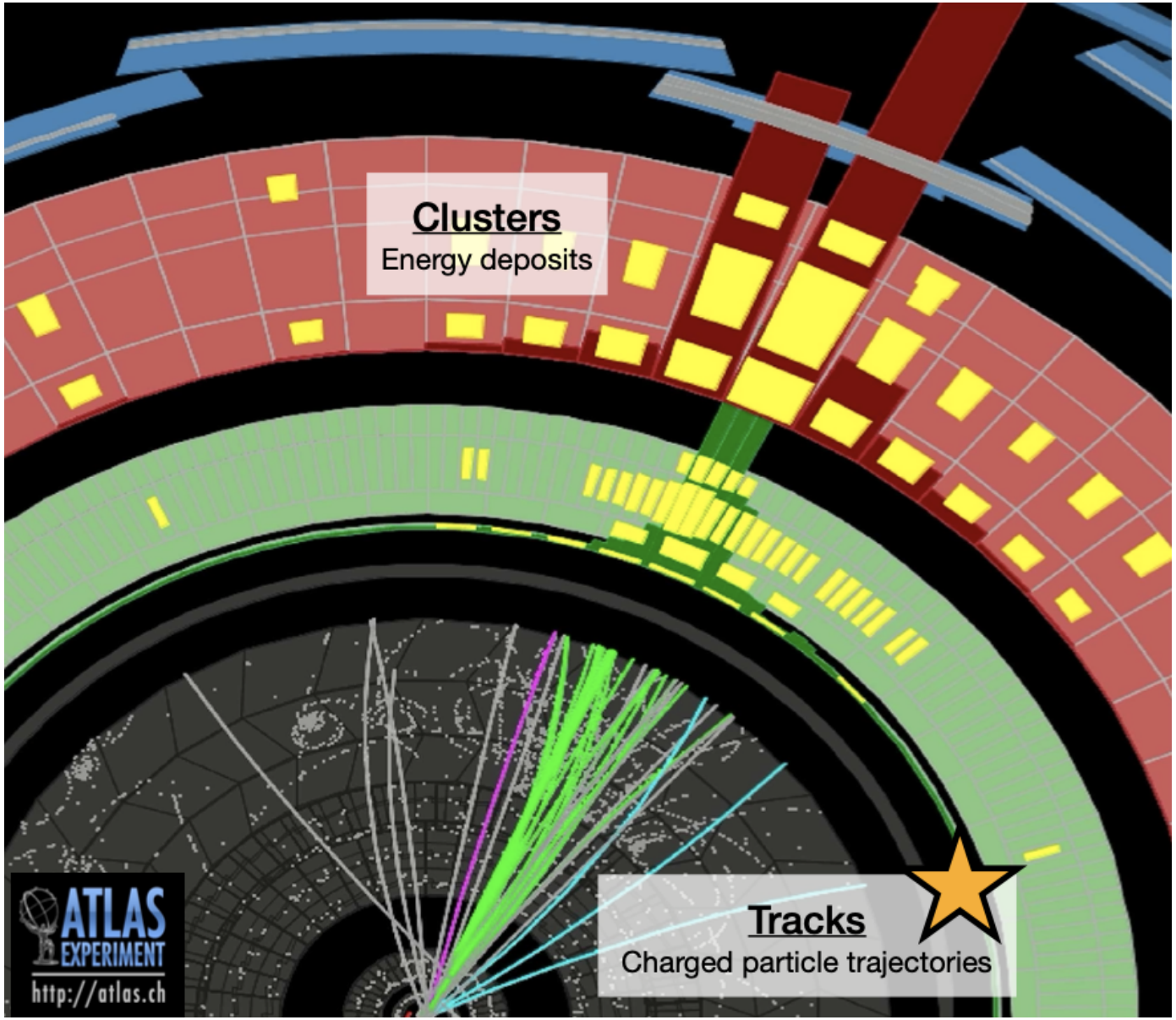


Complex model doing *worse* than the simple one on the training dataset!

Doesn't make sense from the bias / variance picture we built up earlier!
Implies an *optimization problem*.

The European Physical Journal C
Vol 73 3 (2013) 2304

Input representation



Collection of tracks:

$$X_i : i = \{ 1 , \dots , n \}$$

Each track has features:

$$X_i \in \mathbb{R}^m$$

Jet has labels Y

- Quark / gluon tagging: Y ∈ {q, g}
- Higgs tagging: Y ∈ {H, top, QCD}
- Top tagging: Y ∈ {top, QCD}
- b-tagging: Y ∈ {b, c, l}

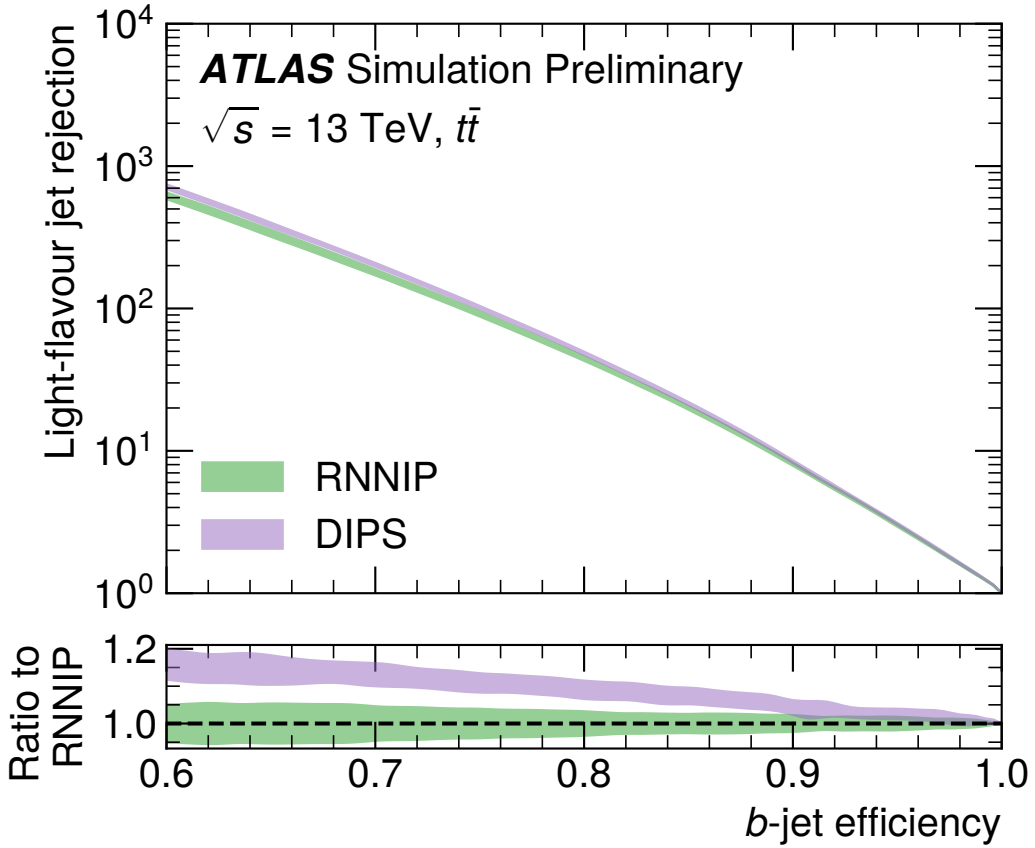
Want: $p(Y \mid X_1, \dots , X_n)$

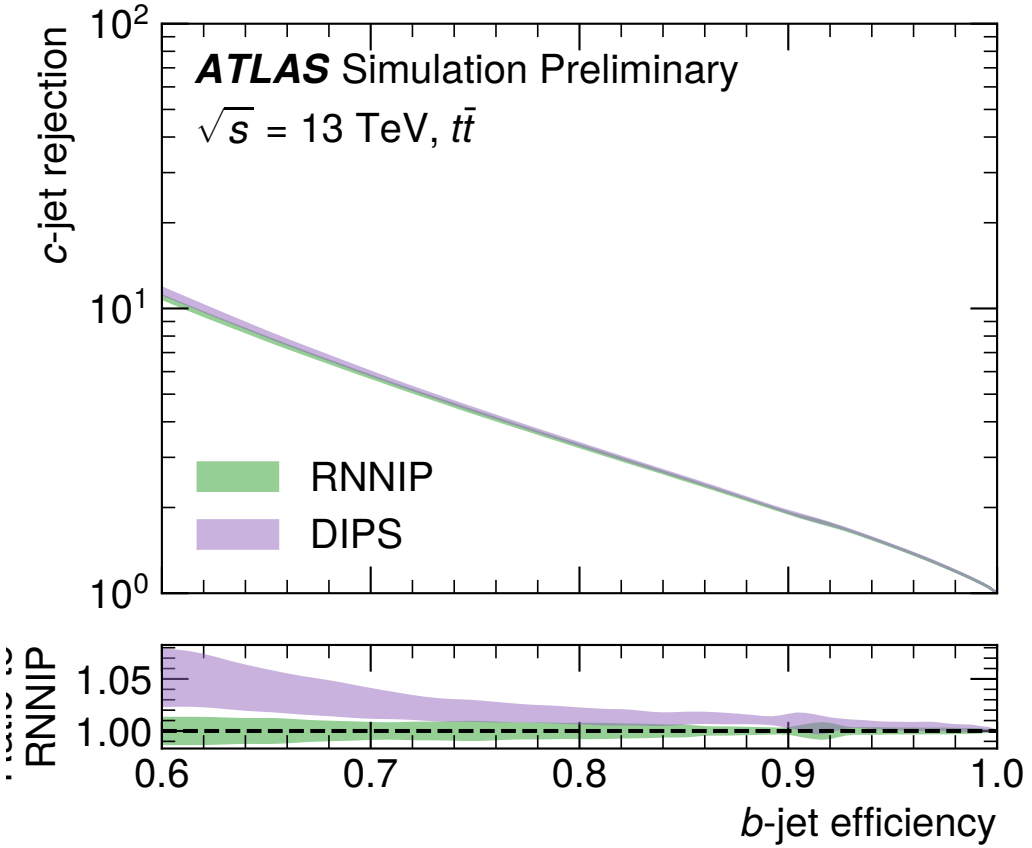
High dimensional problem
n* m ~ O(100)

SLAC

Ensembles

Good way to interpret models!! Let's us *probe* whether experiments are independent of each other





Debugging learning algorithms

Motivating example:

- Anti-spam. You carefully choose a small set of 100 words to use as features. (Instead of using all 50000+ words in English.)
- Bayesian logistic regression, implemented with gradient descent, gets 20% test error, which is unacceptably high.

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

- What to do next?

Fixing the learning algorithm

- Bayesian logistic regression:

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda||\theta||^2$$

- Common approach: Try improving the algorithm in different ways.
  - Try getting more training examples.
  - Try a smaller set of features.
  - Try a larger set of features.
  - Try changing the features: Email header vs. email body features.
  - Run gradient descent for more iterations.
  - Try Newton's method.
  - Use a different value for λ.
  - Try using an SVM.

- This approach might work, but it's very time-consuming, and largely a matter of luck whether you end up fixing what the problem really is.

I.e, perhaps a jet classification ex?

Bia

Diagnostic for bias vs. variance

Better approach:

- Run diagnostics to figure out what the problem is.
- Fix whatever the problem is.

Bayesian logistic regression's test error is 20% (unacceptably high).

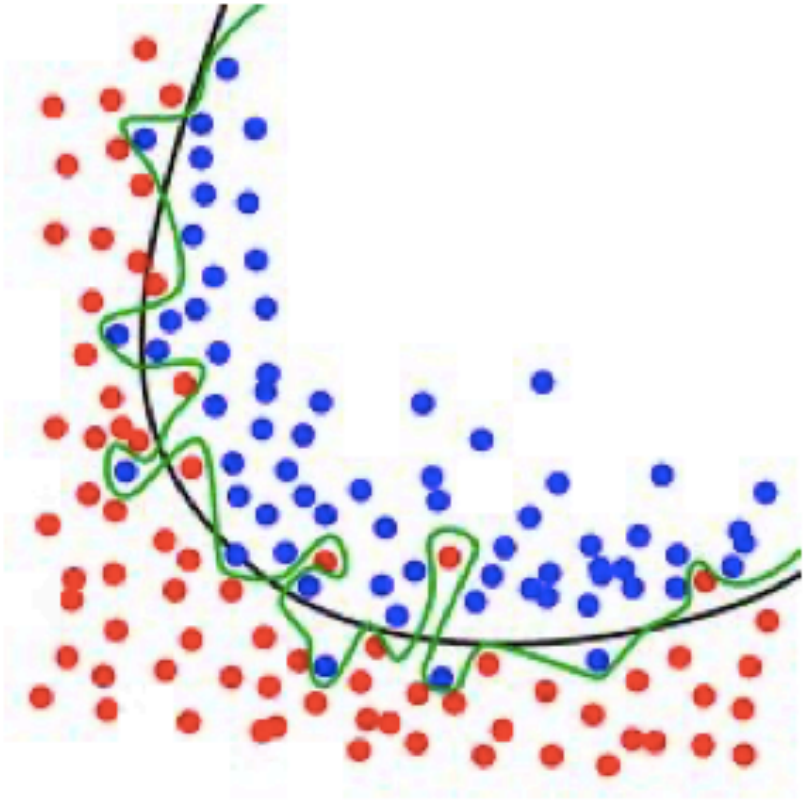
Suppose you suspect the problem is either:

- Overfitting (high variance).
- Too few features to classify spam (high bias).

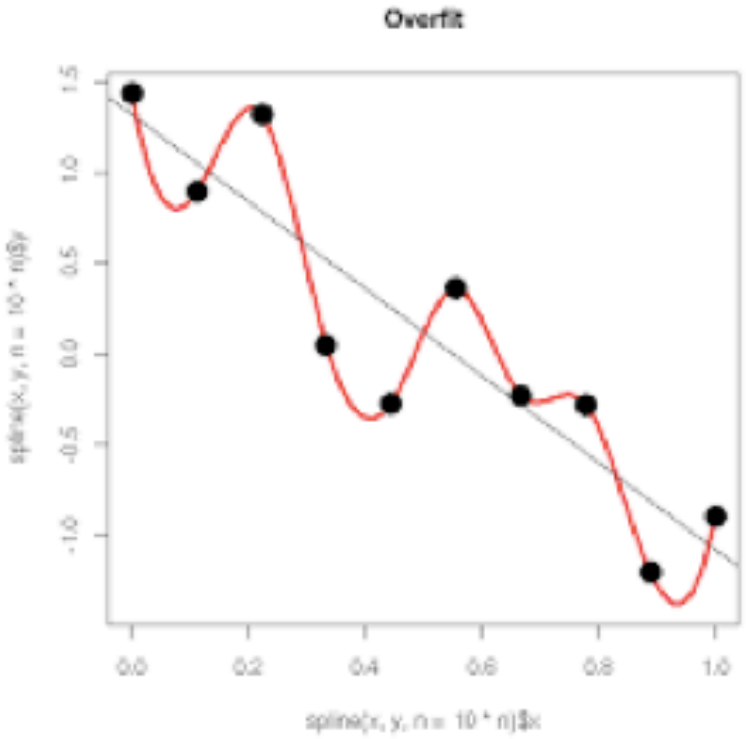
Diagnostic:

- Variance: Training error will be much lower than test error.
- Bias: Training error will also be high.

Overfitting pictures

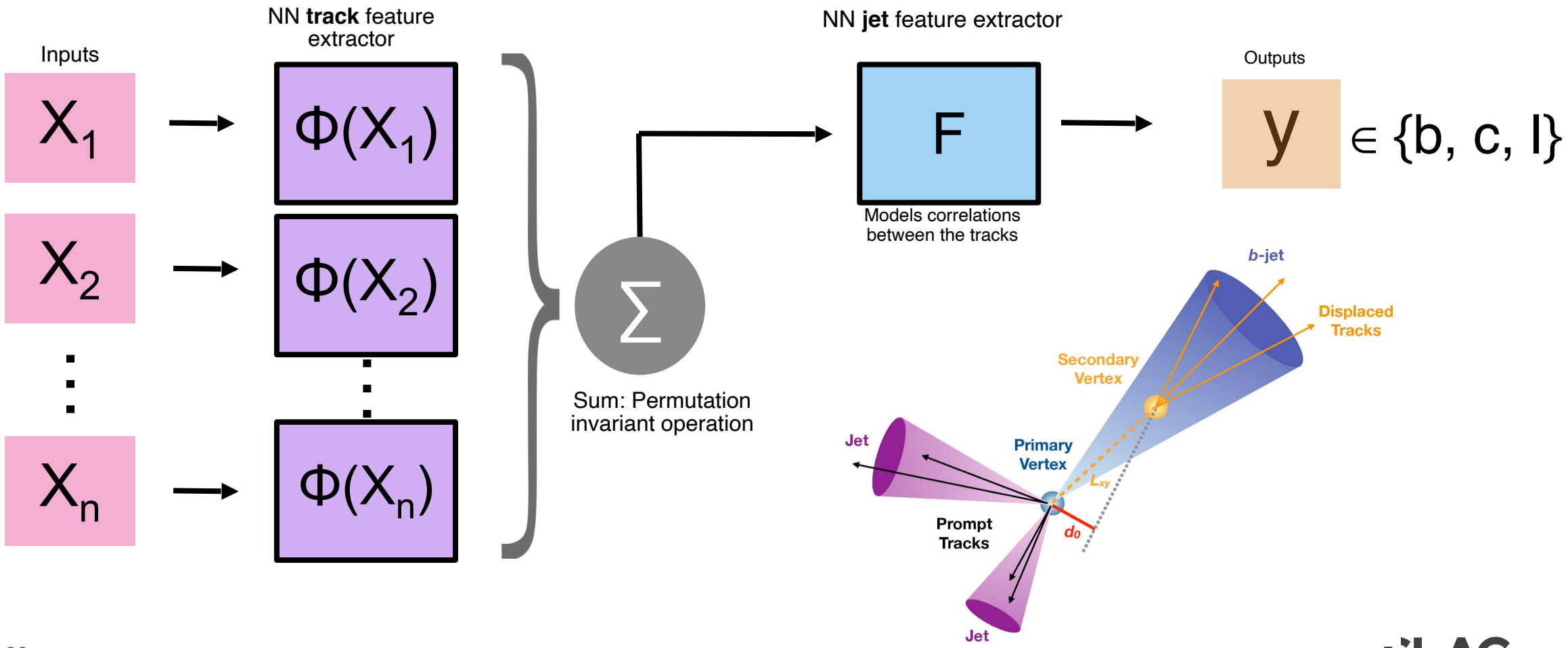


Classification



Regression

Deep Sets: for b-tagging



CNNs

Translational invariance

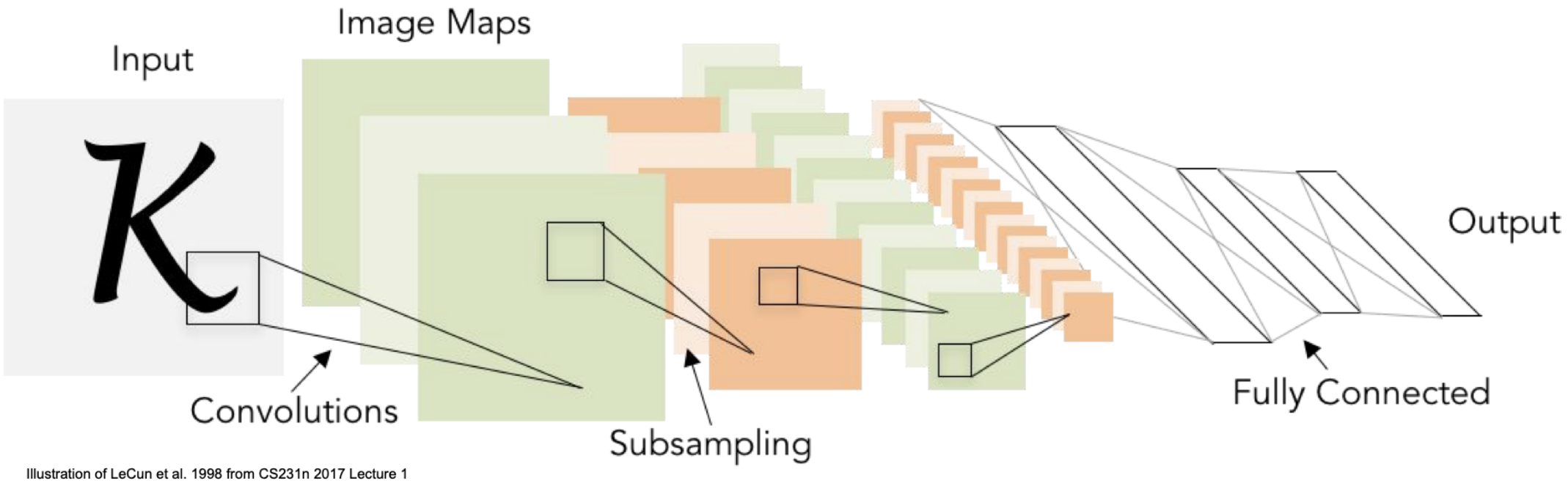


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Strong **inductive bias**: or domain knowledge we inject to efficiently converge to a solution.

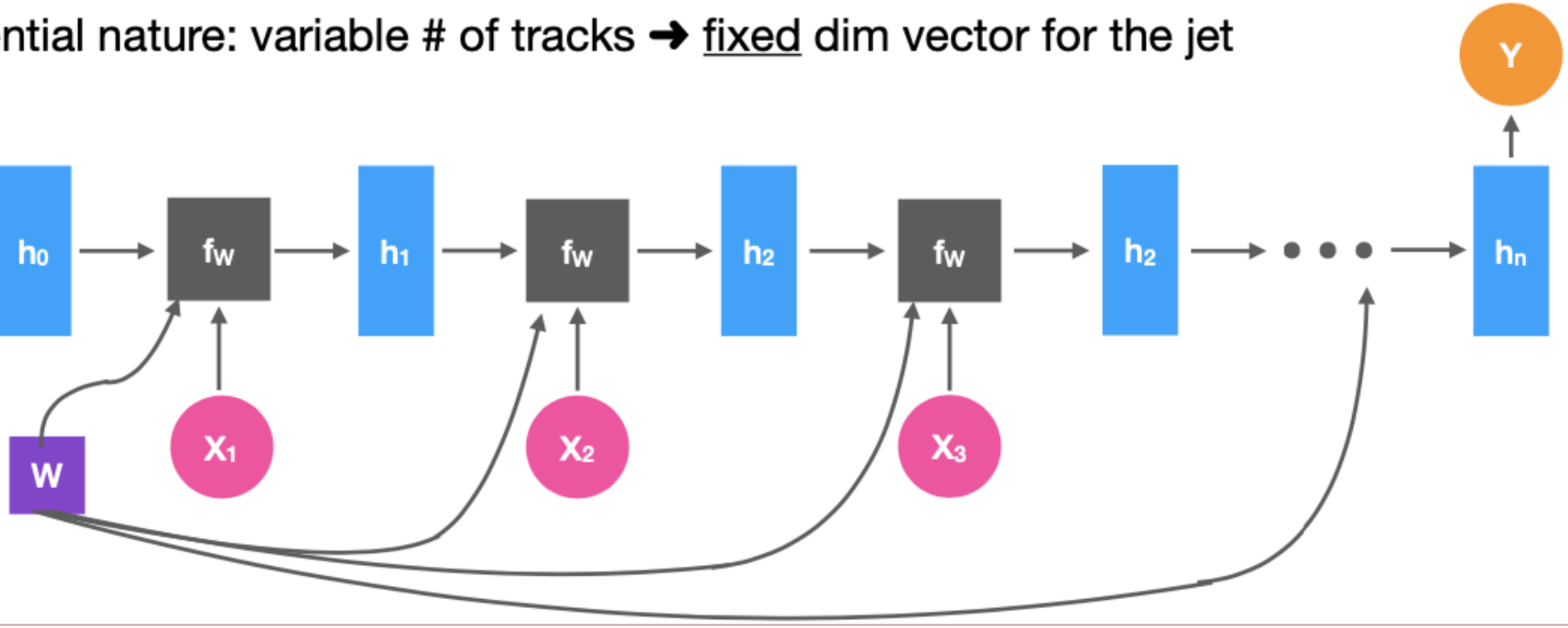
SLAC

Graphic adapted from Stanford
CS 231n Lecture 10: RNNs

Recurrent neural networks

Model the jet as a sequence

- Efficient representation: same tranformation at every time step
- Sequential nature: variable # of tracks ➜ fixed dim vector for the jet

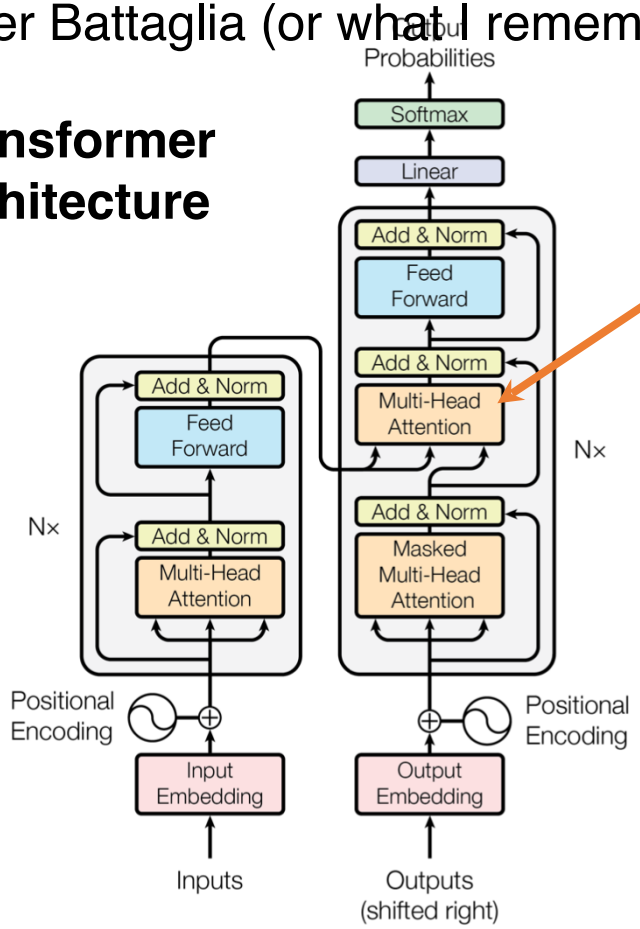


SLAC

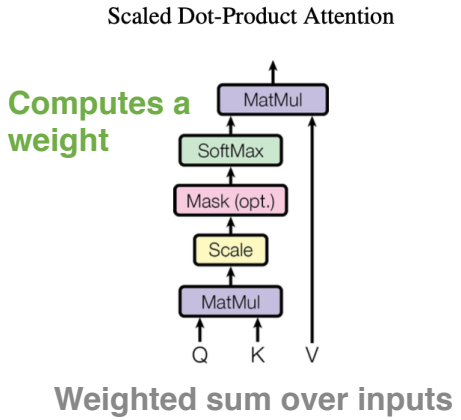
Transformers

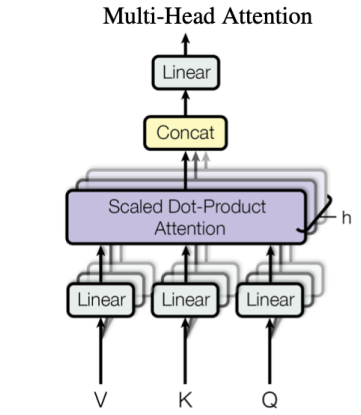
"Any time you have a deep set you can substitute in a transformer as it's always more expressive."
~ Peter Battaglia (or what I remembered of the quote)

Transformer architecture



Replace the Deep Sets Sum operation with a weighted sum (can still be permutation invariant)





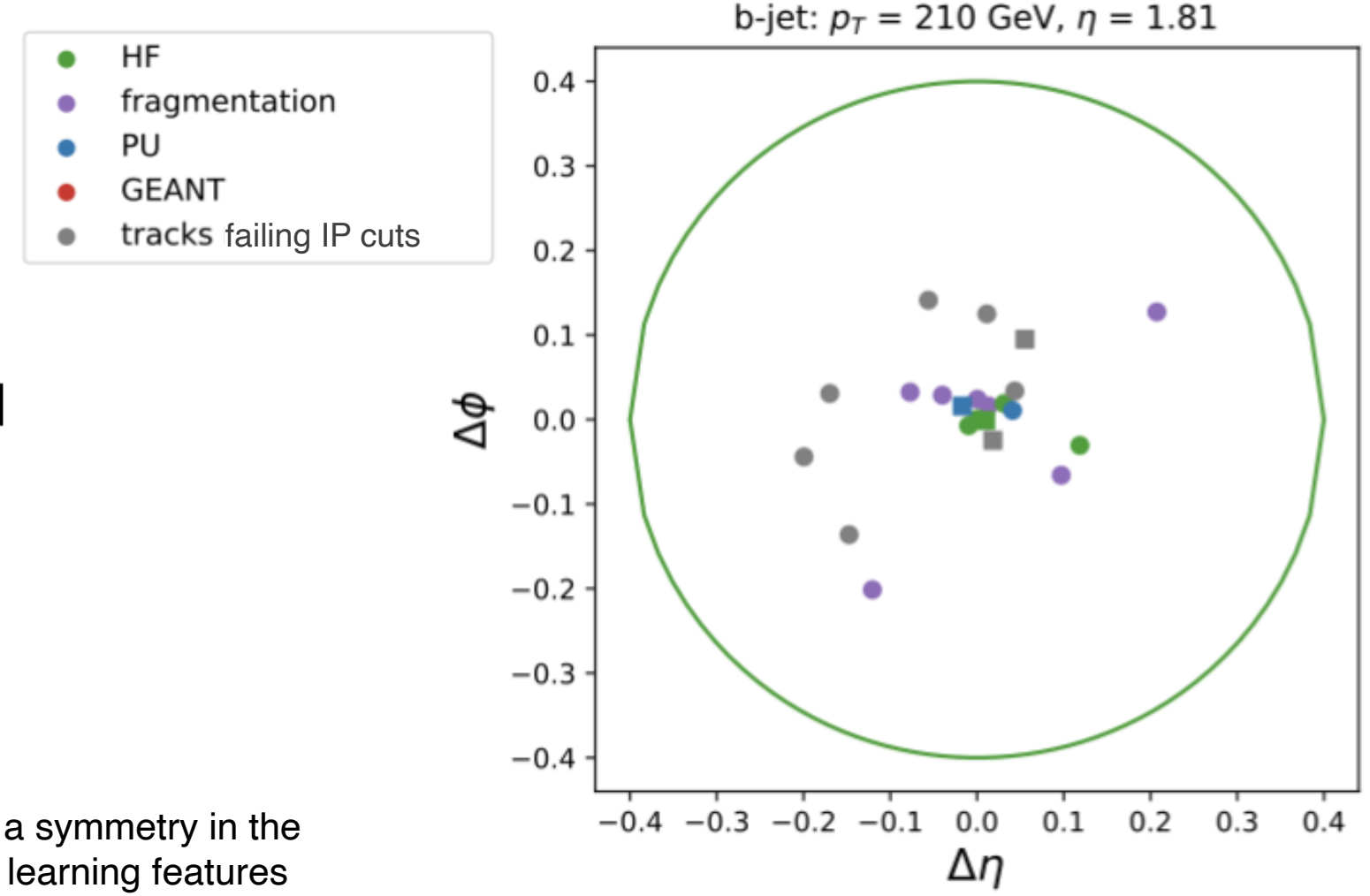
Standard architecture for Natural Language Processing (replacing RNNs): faster training -> better optimization.

SLAC

Input processing: ΔR

Ex: b-tagging input features

$$X_i = \left\{ \begin{array}{l} s_{d0} \\ s_{z0} \\ p_T^{frac} \\ \Delta R \\ \text{IBL hits} \\ \text{PIX1 hits} \\ \text{shared IBL hits} \\ \text{split IBL hits} \\ \text{nPixHits} \\ \text{shared pixel hits} \\ \text{split pixel hits} \\ \text{nSCTHits} \\ \text{shared SCT hits} \\ d_0 \\ z_0 \sin\theta \end{array} \right\}$$



Training with ΔR(trk,jet) encodes a symmetry in the input representation - and makes learning features easier.

SLAC

Saliency Maps

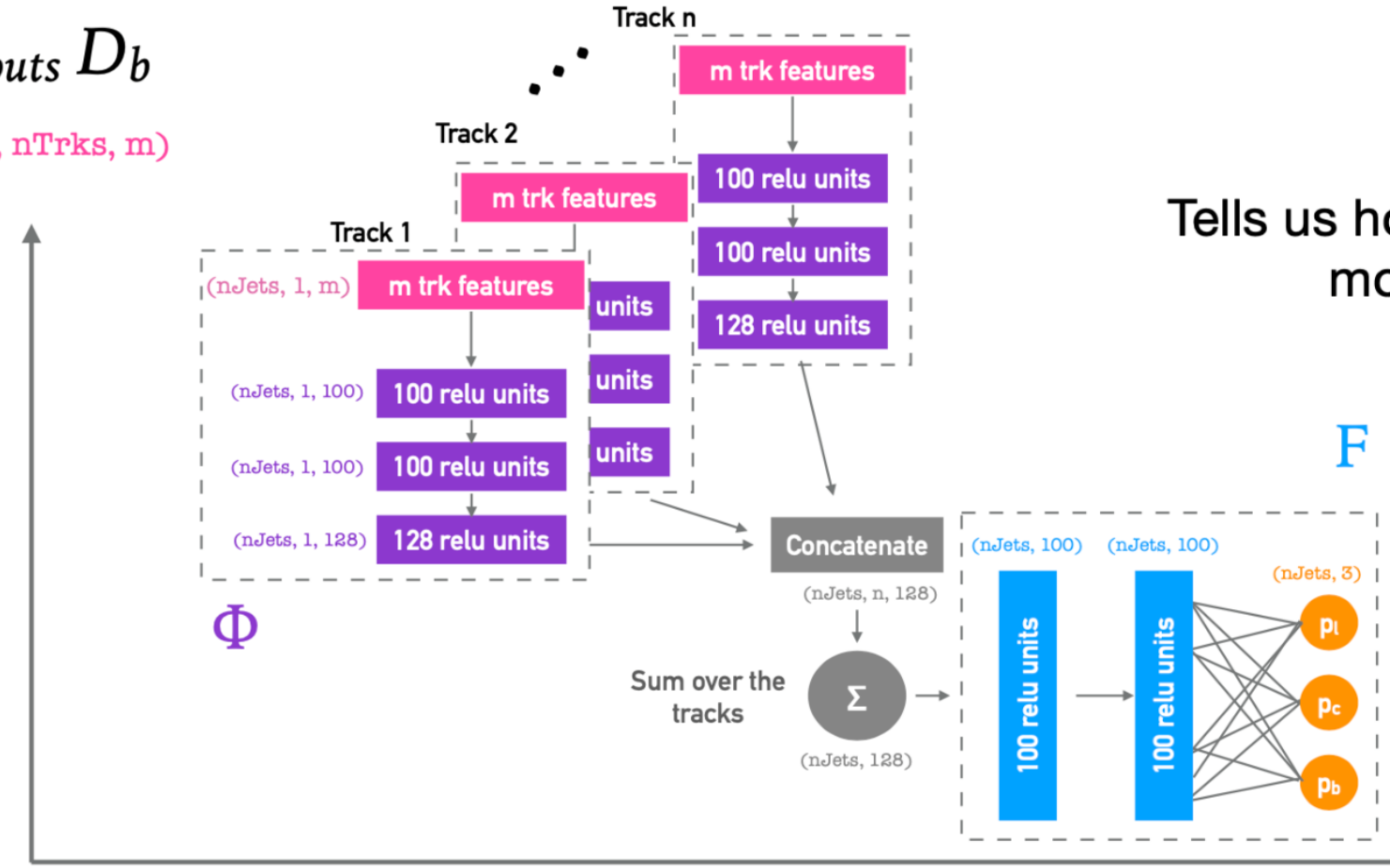
Model diagnostic

Saliency maps

What has DIPS learned about b-jets?

$\nabla_{inputs} D_b$

(nJets, nTrks, m)



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

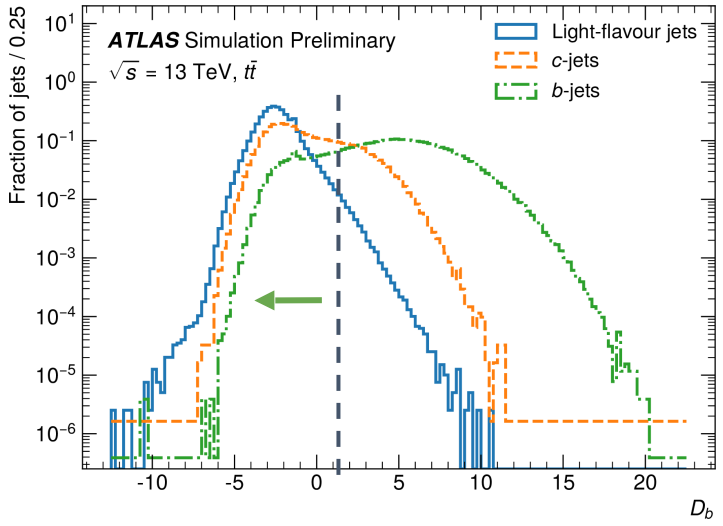
$$D_b = \log \frac{p_b}{f_c p_c + (1 - f_c) p_l}$$

SLAC

Saliency definition

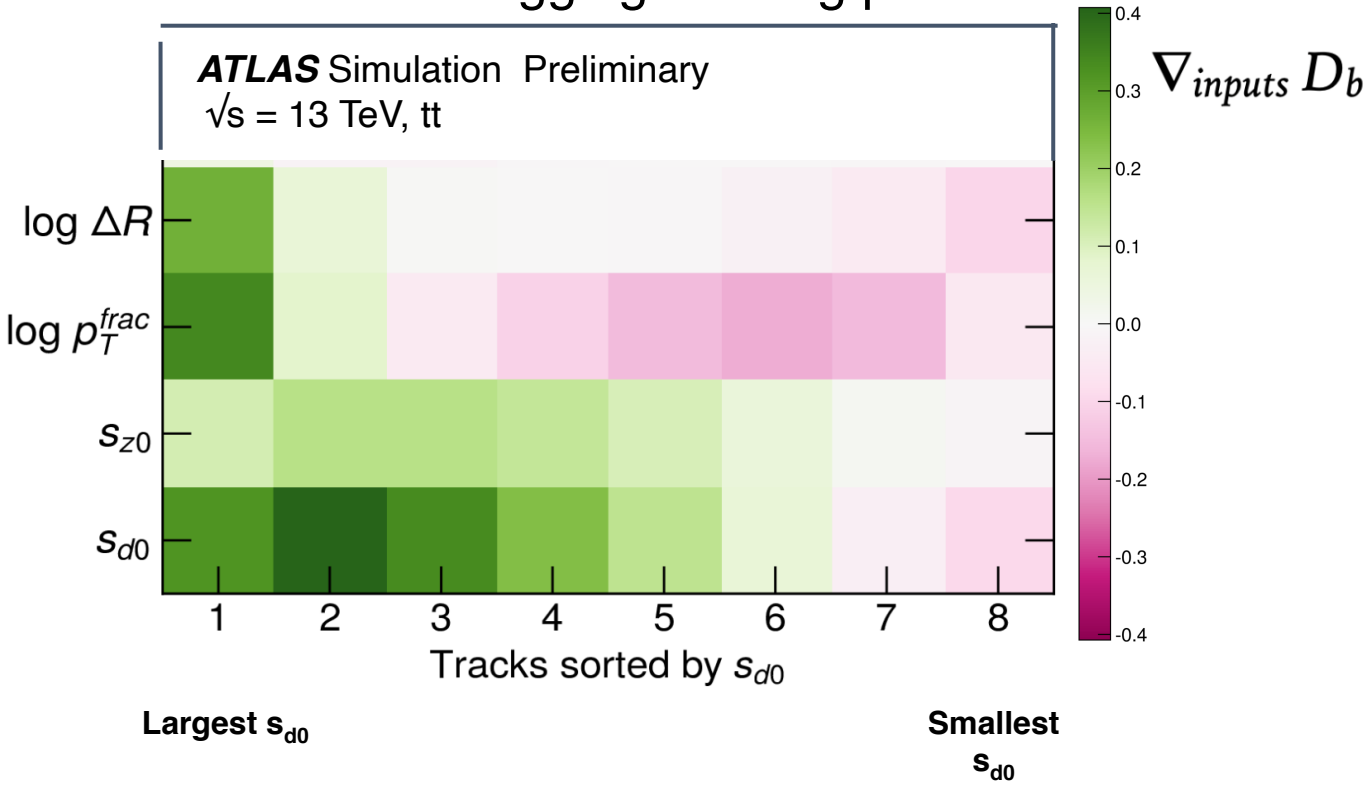
What has DIPS learned about b-jets?

- 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

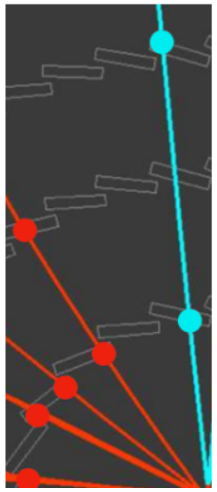


Saliency for DIPS

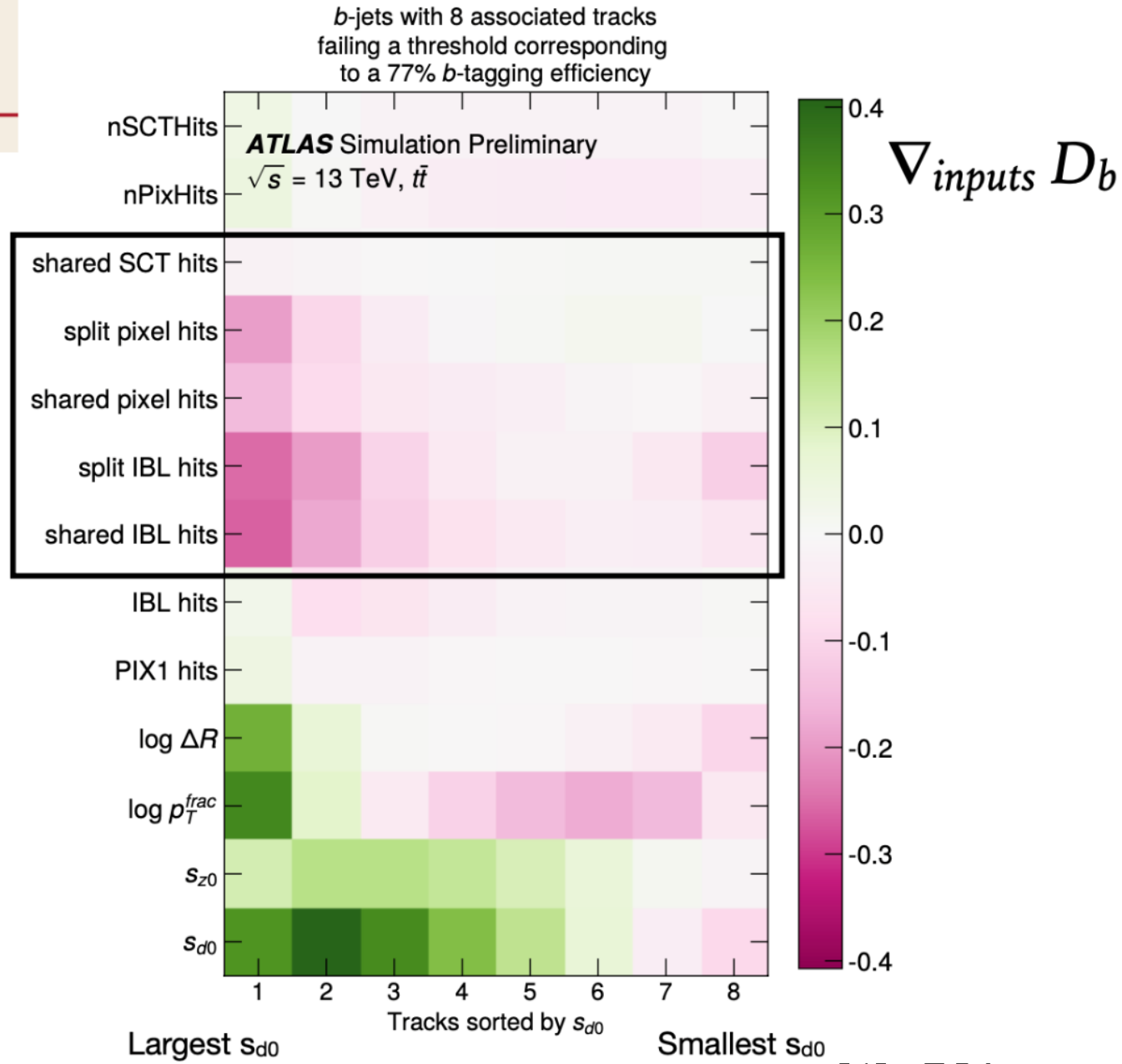
ATL-PHYS-PUB-2020-014

What has DIPS learned about b-jets?

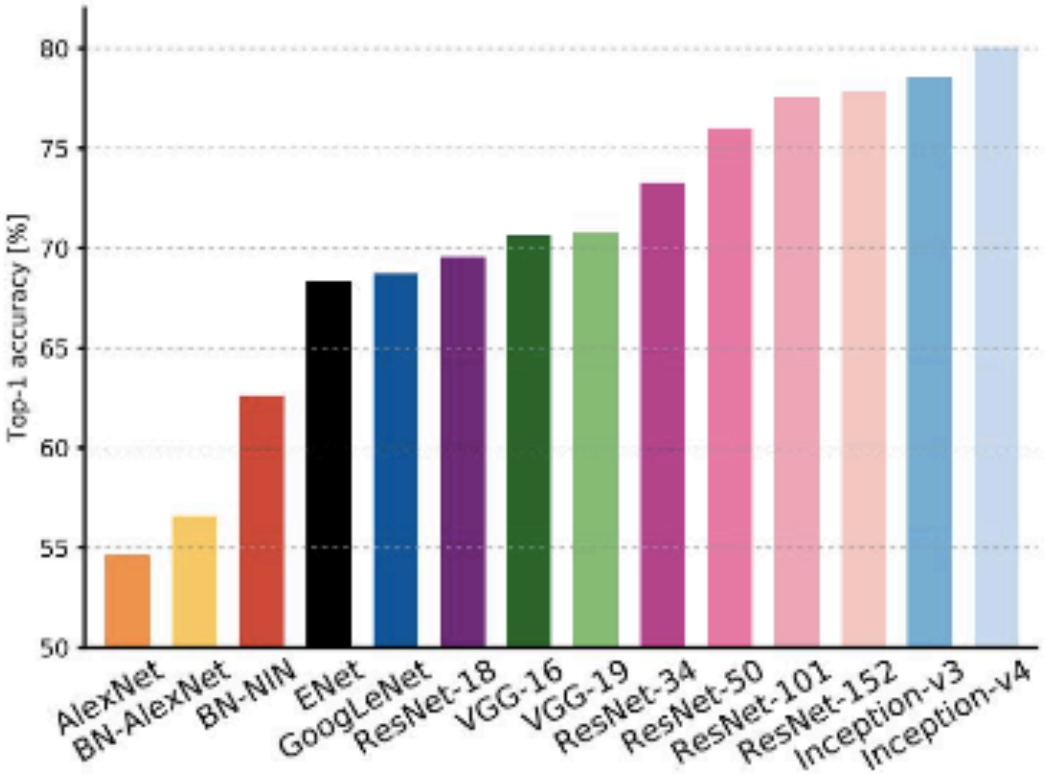
- ✓ Want at least 5 high impact parameter tracks
- ✓ Wants harder leading track - as expected from b-decay
- ✓ Larger opening angle corresponding to geometrical constraint from more displaced tracks
- ✓ Want good quality for displaced tracks

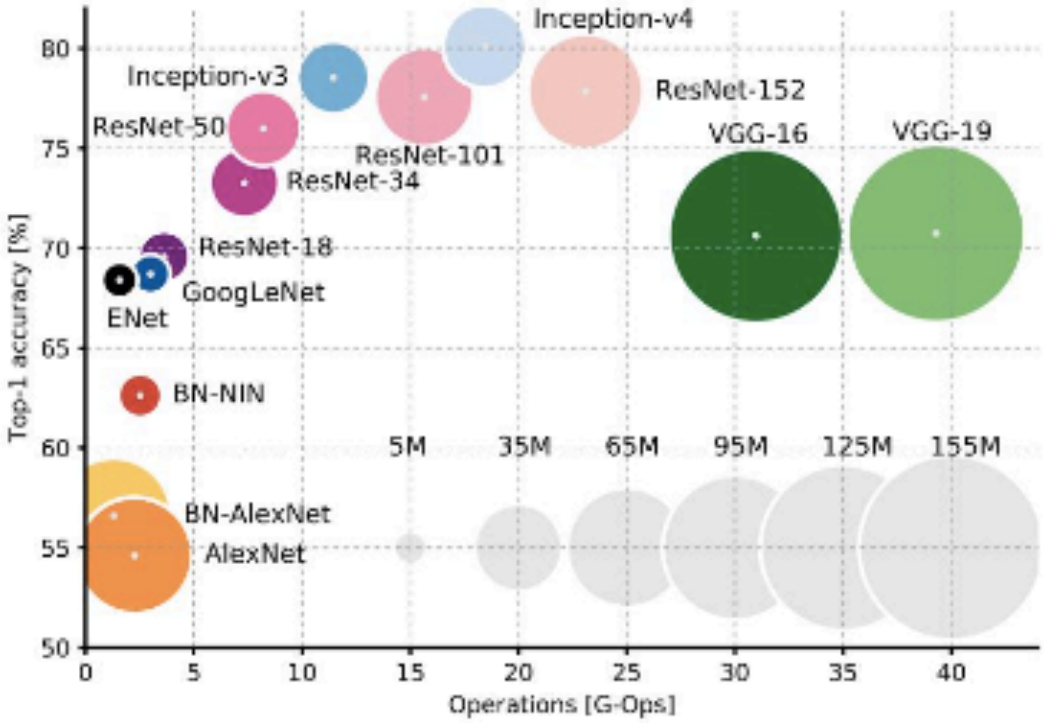


Hit visualization



The Deep Learning Revolution

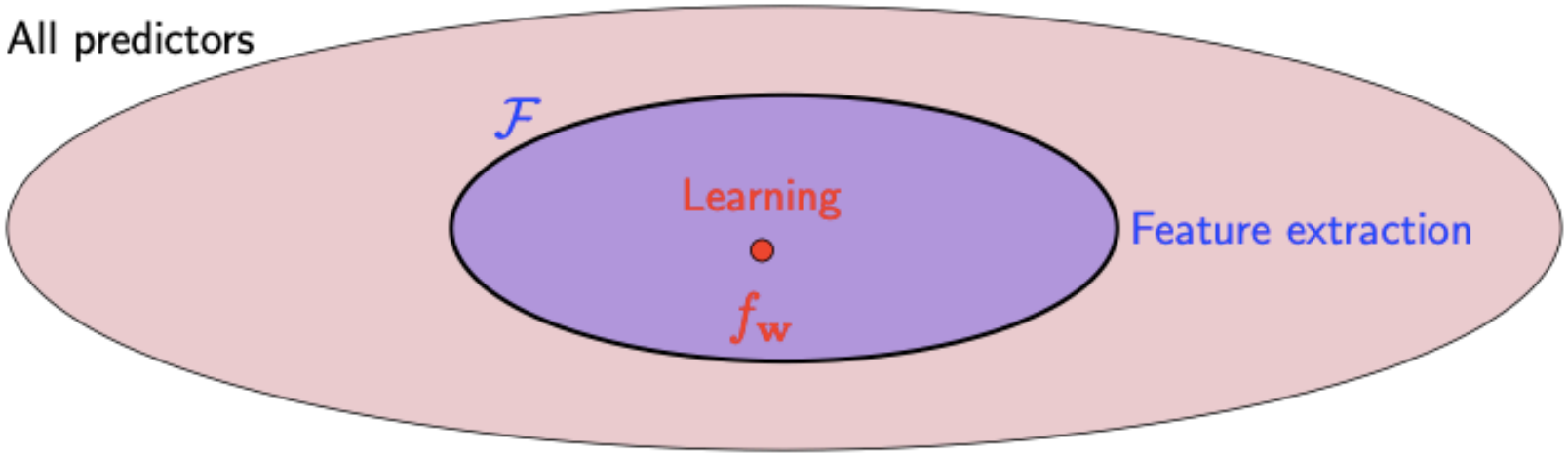




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

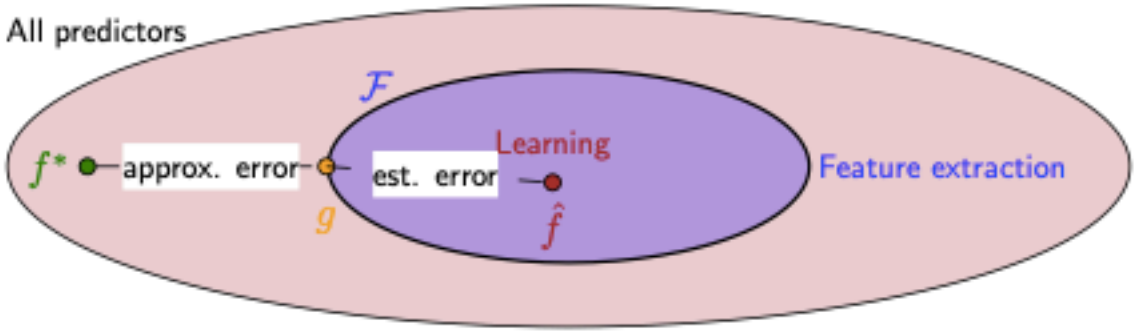
Choosing the right features

Feature extraction + learning



- Feature extraction: set $\mathcal{F}$ based on domain knowledge
- Learning: set $f_{\mathbf{w}} \in \mathcal{F}$ based on data

Effect of hypothesis class size



As the hypothesis class size increases...

Approximation error decreases because:

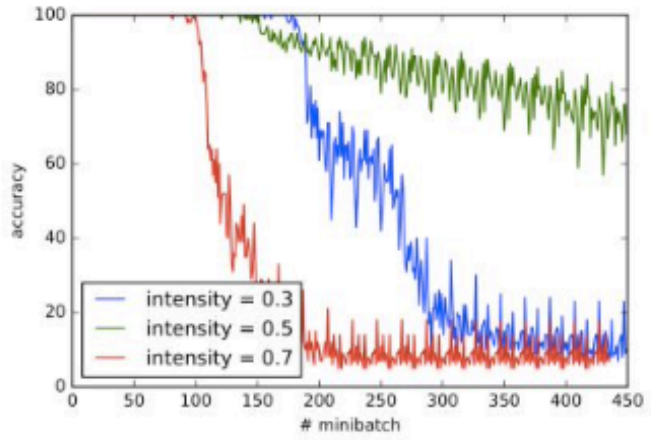
taking min over larger set

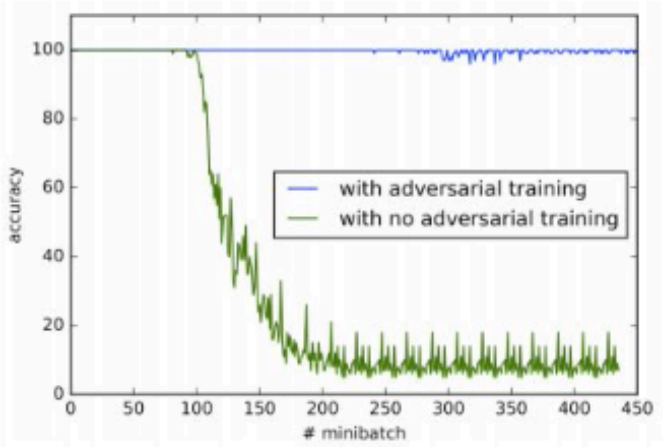
Estimation error increases because:

harder to estimate something more complex

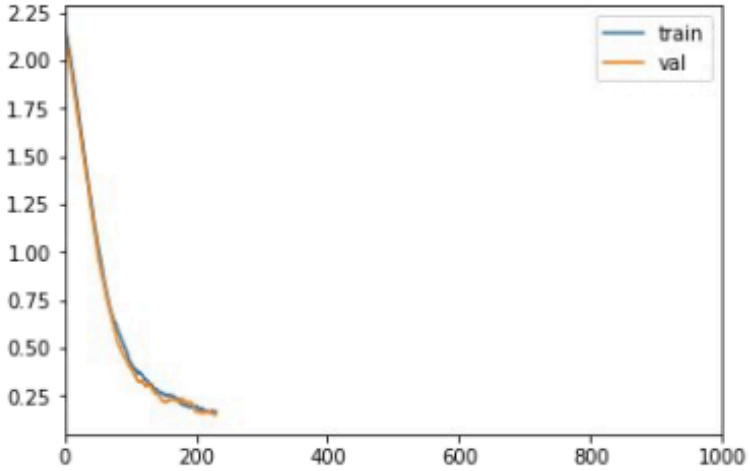
How do we control the hypothesis class size?

Loss curves — what are the problems?

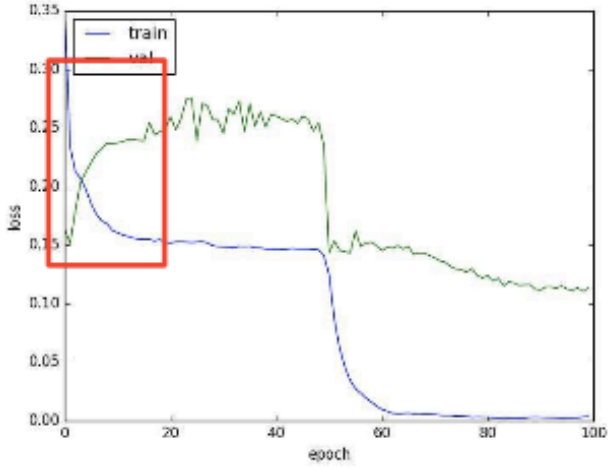




Problem: Not shuffling data, periodical patterns in loss curve



Get nans in the loss after a number of iterations: caused by numerical instability in models



Problem: val set too small, statistics not meaningful

Batch Normalization

Alternative Placements

OG paper

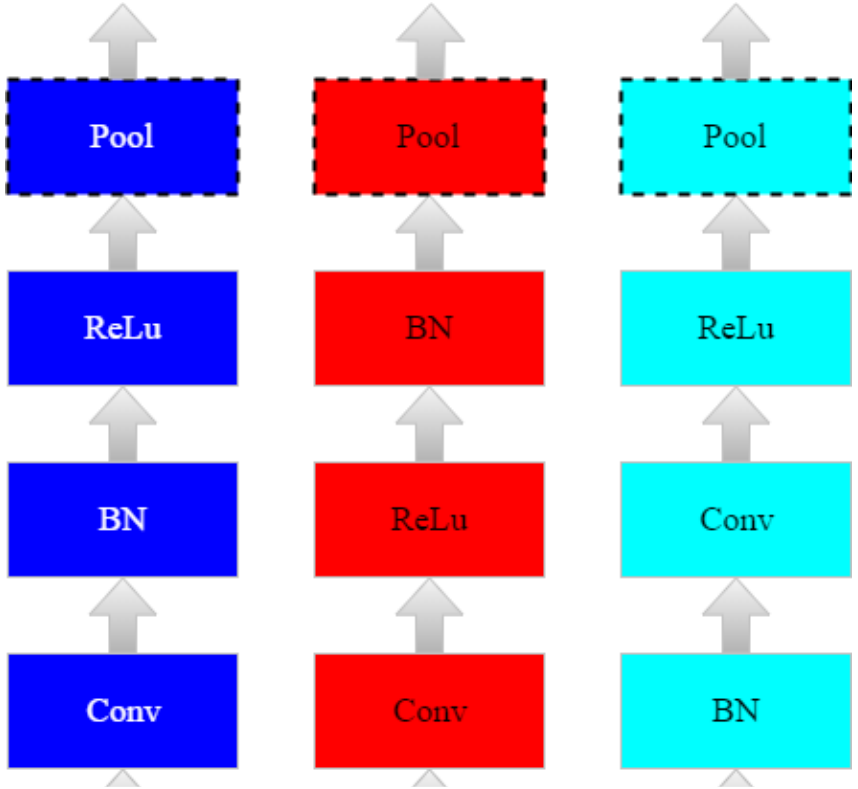
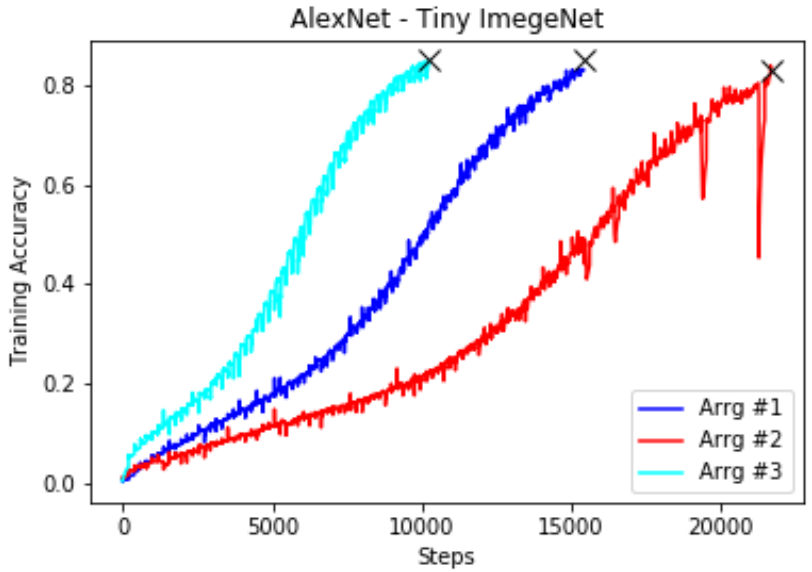
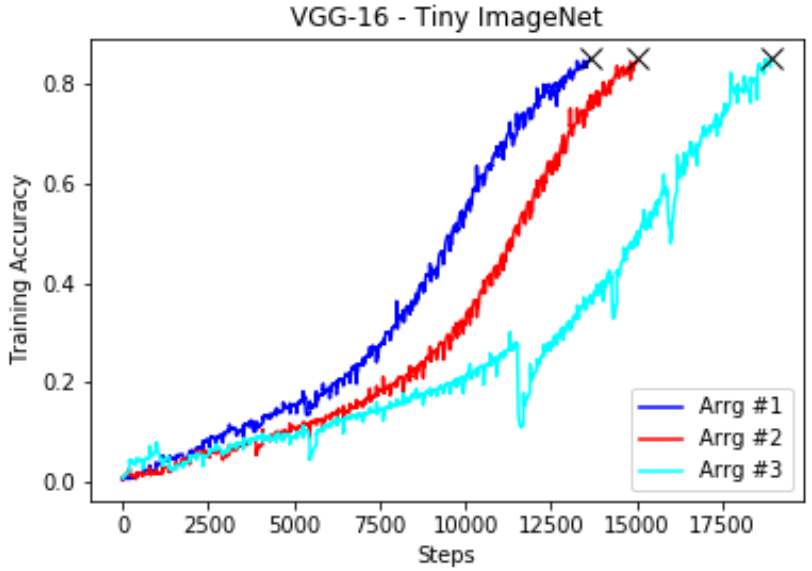


Fig. 1. Different arrangements of layers used in this study. (a) arrangement 1, (b) arrangement 2 and (c) arrangement 3. (The Pooling layer is dashed because it might not always be present after the third layer in these arrangements.)





Double Descent