AD511 Data science and AI prototyping

Machine Learning II

PhD candidate
Yu-Chung Wang

Core Knowledge

ML - Neural Network pipeline

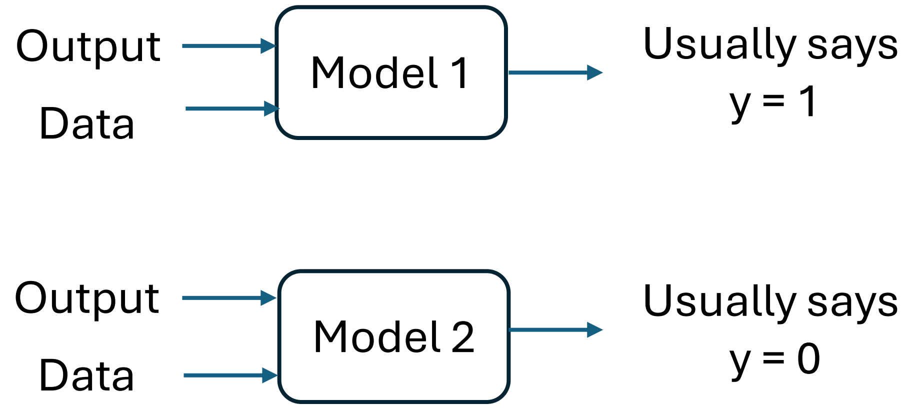
- **Evaluation Metric**
- **Overfitting and underfitting and how do we solve them**
- How to connect to ***probrability and utility***? Model Calibration

Large Language Model

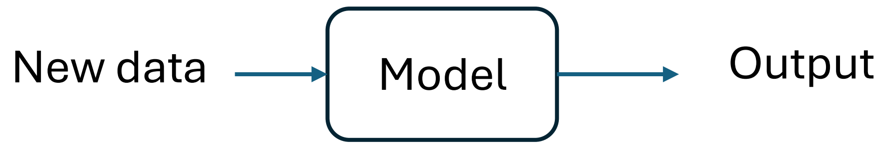
- General Introduction for LLM
- A popular application called Retrieval-Augmented Generation (RAG)

Evaluation is Creation

A binary classification case:



In machine learning, the way you evaluate a model is the way you create it.



- Does this model do a good job at mapping 'New Data → Output'?

  metric
- Is one model better at mapping 'New Data → Output' than another?

  A fair performance measure
- Are the mistakes similar or different? Which is better?

If I've tried 1,000 models, which should I use?

Getting Data for Evaluation

1) **Training set**: to build the model
2) **Validation set**: tune the hyperparameters
3) **Test set**: to estimate how well the model works

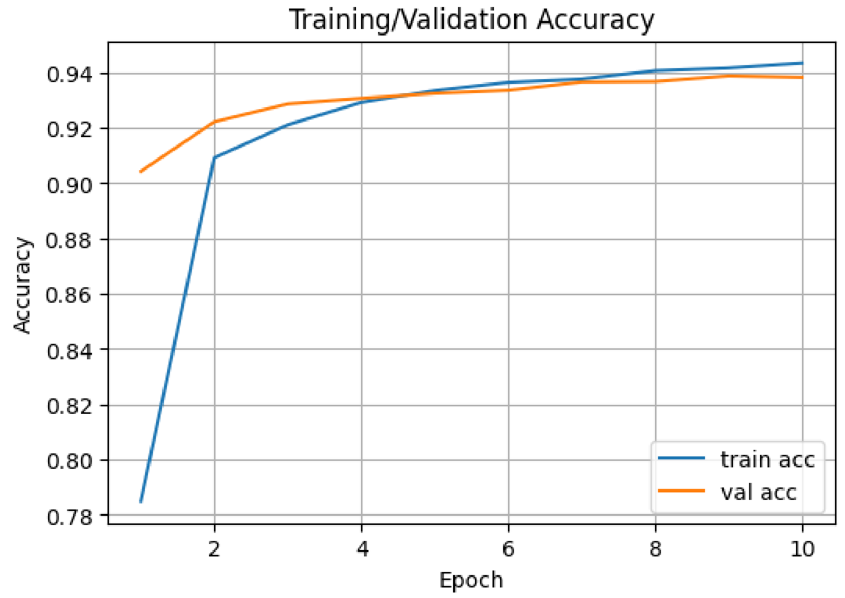
Common Pattern

```
for p in hyperparametersToTry:
    model.fit(trainX, trainY, p)
    accuracies[p] = evaluate(validationY, model.predict(validationX))

bestHyperparameters = max(accuracies, key=lambda k: accuracies[k])

finalModel.fit(
    trainX + validationX,
    trainY + validationY,
    bestHyperparameters
)

estimateOfGeneralizationPerformance = evaluate(
    testY,
    finalModel.predict(testX)
)
```



```
loss: 0.20601396262645721
Accuracy 0.9369000196456909
```

Risk with Evaluation

Failure to Generalize:

1. If you test on the same data you train on, you'll be too optimistic
2. If you evaluate on test data a lot as you're debugging, you'll be too optimistic

Failure to learn the best model you can:

3. If you reserve too much data for testing you might not learn as good a model

We'll get into more detail on how to make the tradeoff

For now:

1. If very little data (100s), maybe up to 50% for validate + test
2. If tons of data (millions+), maybe ten thousand for validate + test
3. Most of the works in famous AI/ML conference used training:60% Validation 20% Testing: 20%

Geoff Hulten

@GeoffHulten · 5060位訂閱者 · 35 部影片

Applying machine learning in the real world. ...顯示更多

linkedin.com/in/geoff-hulten-58136a1 和另外 1 個連結

Types of Mistakes: Confusion Matrix

Assume we have a binary classification model.

		Prediction	
		1	0
Actual	1	True Positive	False Negative
	0	False Positive	True Negative

Actual	Prediction
0	1
0	1
0	1
0	0
0	0
1	0
1	0
1	1
1	1
1	1

		Prediction	
		1	0
Actual	1		
	0		

Basic Evalation Metrics

Actual	Prediction
0	1
0	1
0	1
0	0
0	0
1	0
1	0
1	1
1	1
1	1

		Prediction	
		1	0
Actual	1	**3**	**2**
	0	**3**	**2**

		Prediction	
		1	0
Actual	1	True Positive	False Negative
	0	False Positive	True Negative

Accuracy: What fraction does it get right

$$(\#TP + \#TN)/\#Total$$

Precision: When it says 1, how often is it right

$$\#TP/(\#TP + \#FP)$$

Recall: What fraction of 1s does it get right

$$\#TP/(\#TP + \#FN)$$

False Positive Rate: What fraction of 0s are called 1s

$$\#FP/(\#FP + \#TN)$$

False Negative Rate: What fraction of 1s are called 0s

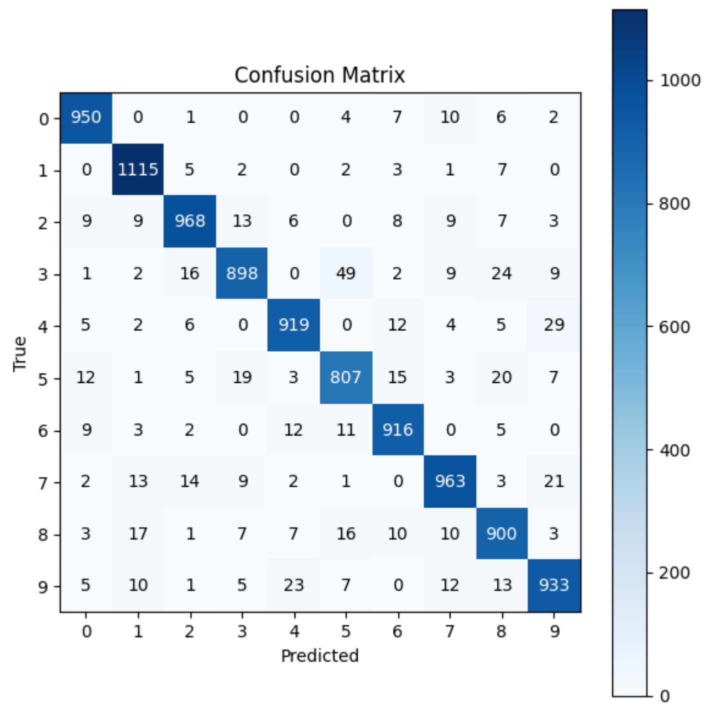
$$\#FN/(\#TP + \#FN)$$

F1 Score: Harmonic mean of Precision and Recall

$$2 \times Precision \times Recall/(Precision + Recall)$$

Basic Evalation Metrics

Hands-on practice last week



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	950	0	1	0	0	4	7	10	6	2
1	0	1115	5	2	0	2	3	1	7	0
2	9	9	968	13	6	0	8	9	7	3
3	1	2	16	898	0	49	2	9	24	9
4	5	2	6	0	919	0	12	4	5	29
5	12	1	5	19	3	807	15	3	20	7
6	9	3	2	0	12	11	916	0	5	0
7	2	13	14	9	2	1	0	963	3	21
8	3	17	1	7	7	16	10	10	900	3
9	5	10	1	5	23	7	0	12	13	933

```
Classification Report:
              precision    recall  f1-score   support

           0     0.9538    0.9694    0.9615       980
           1     0.9514    0.9824    0.9666      1135
           2     0.9500    0.9380    0.9439      1032
           3     0.9423    0.8891    0.9149      1010
           4     0.9455    0.9358    0.9406       982
           5     0.8997    0.9047    0.9022       892
           6     0.9414    0.9562    0.9487       958
           7     0.9432    0.9368    0.9400      1028
           8     0.9091    0.9240    0.9165       974
           9     0.9265    0.9247    0.9256      1009

    accuracy                         0.9369     10000
   macro avg     0.9363    0.9361    0.9361     10000
weighted avg     0.9369    0.9369    0.9368     10000
```

How about multiple classification?

For example, For class 1

Prediction

Actual	Is 1	Is Not 1
Is 1	True Positive	False Negative
Is Not 1	False Positive	True Negative

Actual	Is 1	Is Not 1
Is 1	1115	20
Is Not 1	57	

$$\text{Precision} = \frac{1115}{1115 + 57} \qquad \text{Recall} = \frac{1115}{1115 + 20}$$

Type	Calculation Method	Suitable Scenario
Macro Average	Take the average of the metric across all classes (each class has equal weight)	When all classes are equally important
Weighted Average	Compute the weighted average based on the number of samples in each class	More appropriate when classes are imbalanced

Exercise 8.

		Prediction	
		Is Cancer	Is Not
Actual	Is Cancer	2	98
	Is Not	1	899

- The accuracy is quite **high (90.1%)**, which looks good at first glance
- The model almost failed to identify actual cancer patients since the recall is only **2%.** 100 patients, 98 cancer cases were wrongly classified as healthy.
- The precision is okay **(66.7%)**, meaning that when the model does predict cancer, it is usually correct.
  However, it is so conservative that it rarely predicts anyone as having cancer.

Basic Evalation Metrics

In the regression task, we are supposed to predict the target variable which is in the form of continuous values

$$\mathrm{MAE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} |\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j}|$$

$$\mathrm{MSE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} (\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j})^2$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{\mathrm{j}=1}^{\mathrm{N}} (\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j})^2}{\mathrm{N}}}$$

$$R^2 = 1 - \frac{\sum_{j=1}^{n} (y_j - \hat{y}_j)^2}{\sum_{j=1}^{n} (y_j - \bar{y})^2}$$

Types of Mistakes: Confusion Matrix

Assume we have a binary classification model.

		Prediction	
		1	0
Actual	1	True Positive	False Negative
	0	False Positive	True Negative

How do we apply utility here?

		Prediction	
		1	0
Actual	1	$$$	-$
	0	-$	$

Actual	Prediction
0	1
0	1
0	1
0	0
0	0
1	0
1	0
1	1
1	1
1	1

		Prediction	
		1	0
Actual	1		
	0		

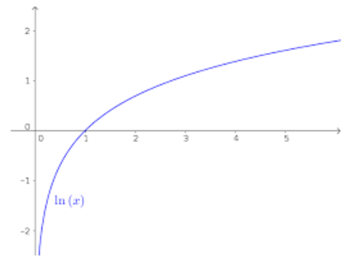
Evaluation Metrics for LLM

Three classic metrics used to evaluate language models and text generation systems:

- Perplexity
- Bilingual Evaluation Understudy (BLEU)
- BERTScore

These capture linguistic fluency, precision, and semantic similarity from different perspectives.

Perplexity



Purpose: Measures a language model's *predictive ability.*

"How uncertain is the model about predicting the next word?"

The cat is on the……..mat

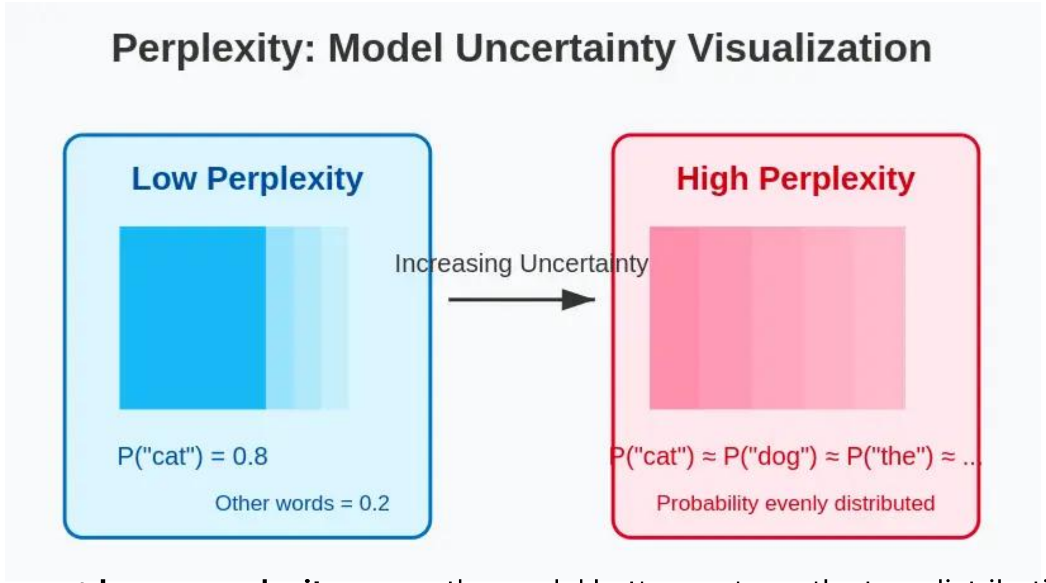
Given a sentence or test corpus,

$$W = (w_1, w_2, ..., w_N)$$

And the model estimates the conditional probability for each word as:

$$P(w_i \mid w_{<i})$$

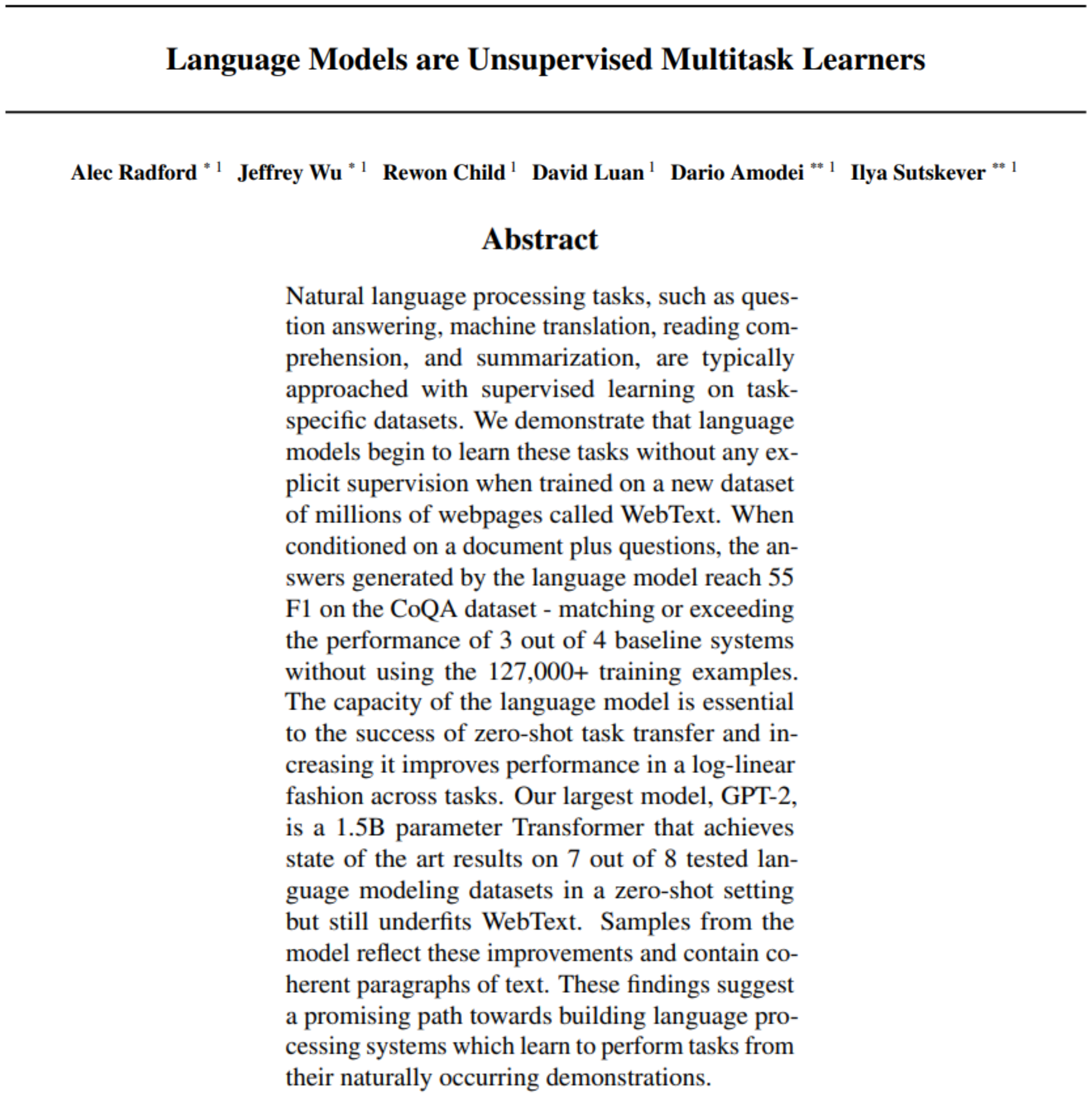
$$\text{Perplexity}(W) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_{<i})\right)$$


Perplexity: Model Uncertainty Visualization
Low Perplexity
High Perplexity
Increasing Uncertainty
P("cat") = 0.8
Other words = 0.2
P("cat") ≈ P("dog") ≈ P("the") ≈ ...
Probability evenly distributed

→ **Lower perplexity** means the model better captures the true distribution of the language.

Cite: https://www.analyticsvidhya.com/blog/2025/04/perplexity-metric-for-llm-evaluation/

Many papers evaluated model performance using perplexity.



Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

Ilya Sutskever (OpenAI co-founder)

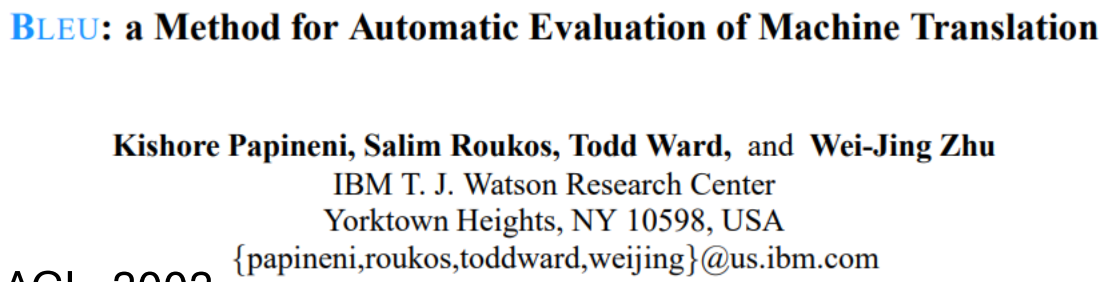
Bilingual Evaluation Understudy (BLEU)

Purpose: To evaluate how closely a model’s generated text matches human reference texts based on overlapping n-grams.

It measures how many *n-grams* (continuous sequences of *n* words) in the candidate sentence also appear in the reference sentence.

Example:
Candidate: *the cat is on the mat*
Reference: *there is a cat on the mat*

1-gram (single words) overlap: Precision = 5/6
2-gram (two-word sequences) overlap: Precision = 2/5



BLEU: a Method for Automatic Evaluation of Machine Translation

Kishore Papineni, Salim Roukos, Todd Ward, and **Wei-Jing Zhu**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{papineni,roukos,toddward,weijing}@us.ibm.com

ACL, 2002

BLEU

Brevity Penalty (BP)

- BLEU penalizes overly short sentences.

$$BP = \begin{cases} 1, & c > r \\ e^{(1-r/c)}, & c \leq r \end{cases}$$

c: candidate sentence
r: reference sentence

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

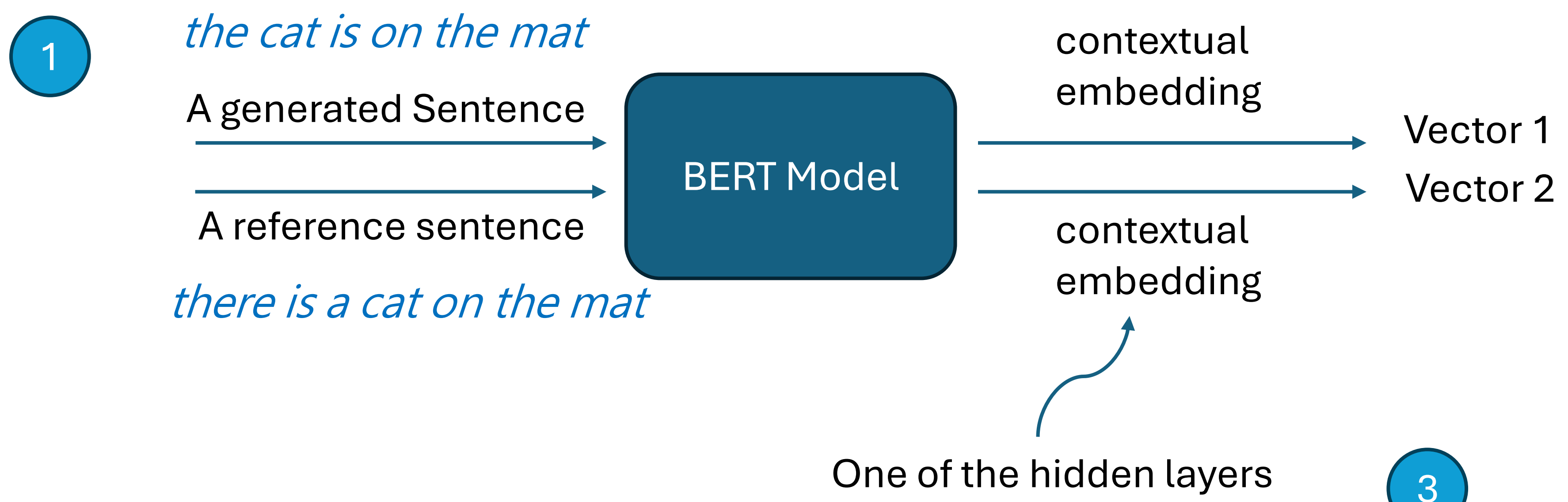
precision for n-grams

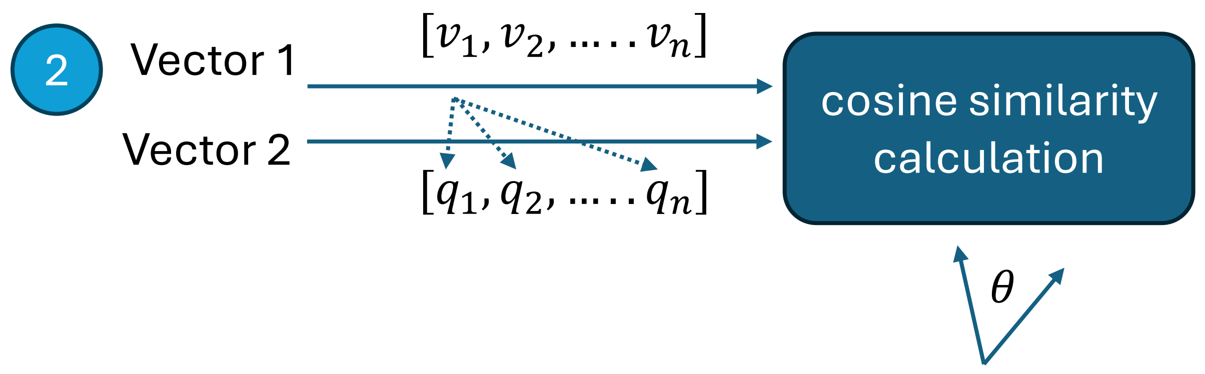
weight for each n-gram (typically $w_n = 1/\mathrm{N}$)

- BLEU score increases with higher n-gram overlap between generated and reference texts.
- Higher is better in most case, but it reflects surface similarity rather than true semantic quality.

ACL, 2002

BERTScore




2
Vector 1
Vector 2
$[v_1, v_2, \ldots\ldots v_n]$
$[q_1, q_2, \ldots\ldots q_n]$
cosine similarity calculation
θ

3

$$\text{Precision} = \frac{1}{|C|} \sum_{x \in C} \max_{y \in R} \cos(x, y)$$

$$\text{Recall} = \frac{1}{|R|} \sum_{y \in R} \max_{x \in C} \cos(x, y)$$

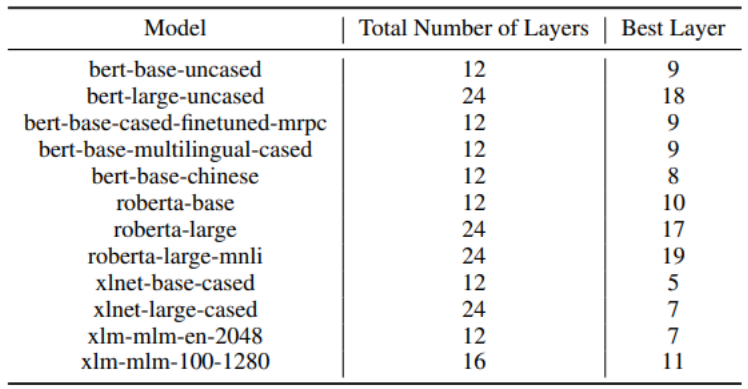
$$\text{BERTScore} = F1(\text{Precision}, \text{Recall})$$

BERTScore

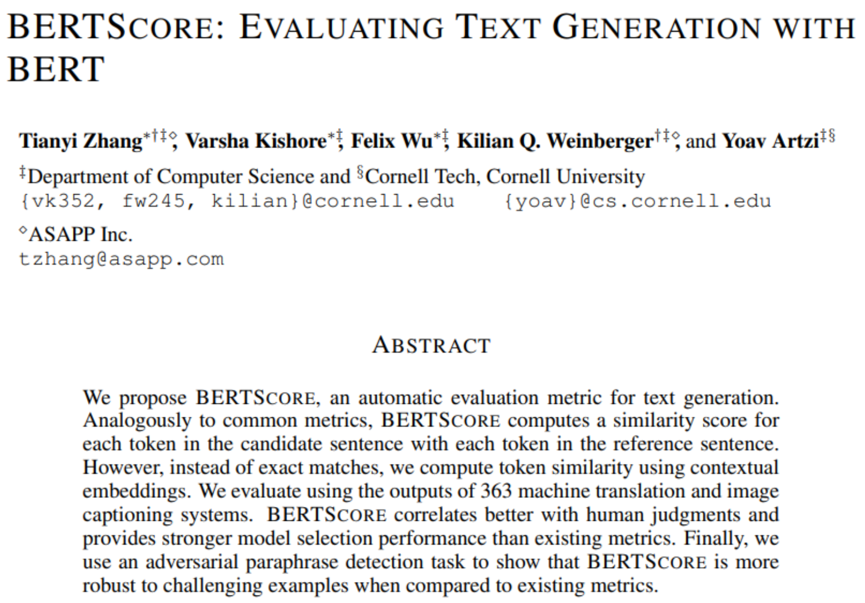
Why do they choose BERT model?

During BERT training, each layer learns different levels of language features

Which layer should we choose?



Model	Total Number of Layers	Best Layer
bert-base-uncased	12	9
bert-large-uncased	24	18
bert-base-cased-finetuned-mrpc	12	9
bert-base-multilingual-cased	12	9
bert-base-chinese	12	8
roberta-base	12	10
roberta-large	24	17
roberta-large-mnli	24	19
xlnet-base-cased	12	5
xlnet-large-cased	24	7
xlm-mlm-en-2048	12	7
xlm-mlm-100-1280	16	11



BERTScore: Evaluating Text Generation with BERT

Tianyi Zhang[*†‡◇], **Varsha Kishore**[*‡], **Felix Wu**[*‡], **Kilian Q. Weinberger**[†‡◇], and **Yoav Artzi**[‡§]

[‡]Department of Computer Science and [§]Cornell Tech, Cornell University
{vk352, fw245, kilian}@cornell.edu   {yoav}@cs.cornell.edu

[◇]ASAPP Inc.
tzhang@asapp.com

Abstract

We propose BERTScore, an automatic evaluation metric for text generation. Analogously to common metrics, BERTScore computes a similarity score for each token in the candidate sentence with each token in the reference sentence. However, instead of exact matches, we compute token similarity using contextual embeddings. We evaluate using the outputs of 363 machine translation and image captioning systems. BERTScore correlates better with human judgments and provides stronger model selection performance than existing metrics. Finally, we use an adversarial paraphrase detection task to show that BERTScore is more robust to challenging examples when compared to existing metrics.

ICLR 2020

- BERTScore uses contextual embeddings from intermediate BERT layers to measure semantic similarity between texts
- It compares tokens based on their meaning in context rather than surface word overlap.

Overfitting and Underfitting

Bias and Variance

- Bias – error caused because the model can not represent the concept
- Variance – error caused because the learning algorithm overreacts to small changes (noise) in the training data

- Bias^2 + Variance + Noise = Total Error

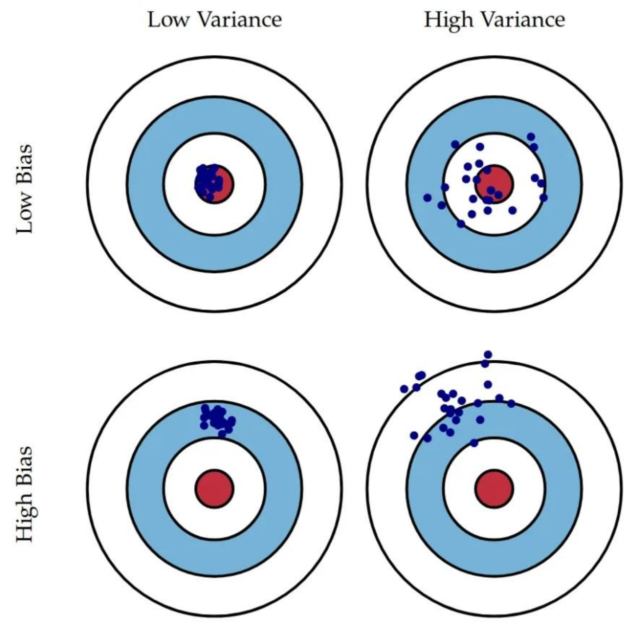
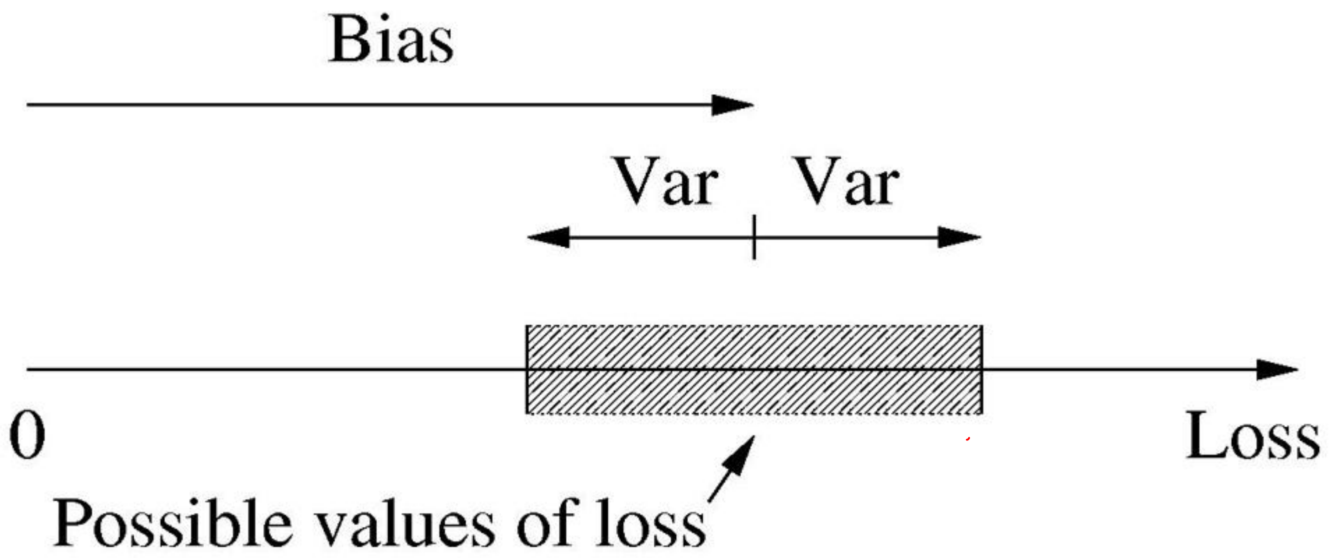


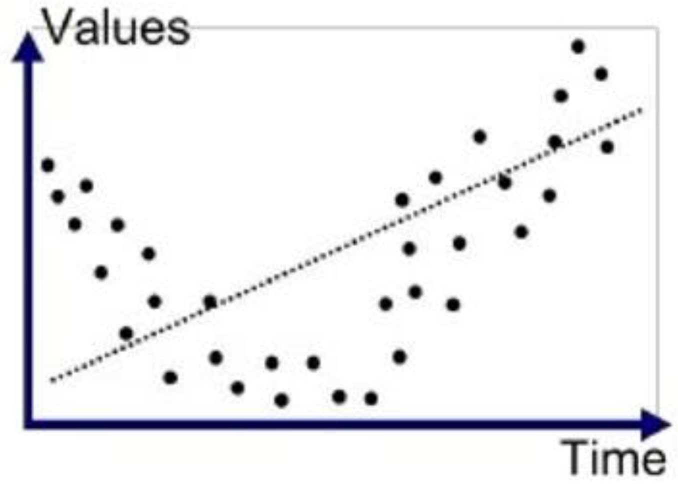
Fig. 1 Graphical illustration of bias and variance.

Another way to think about Bias & Variance

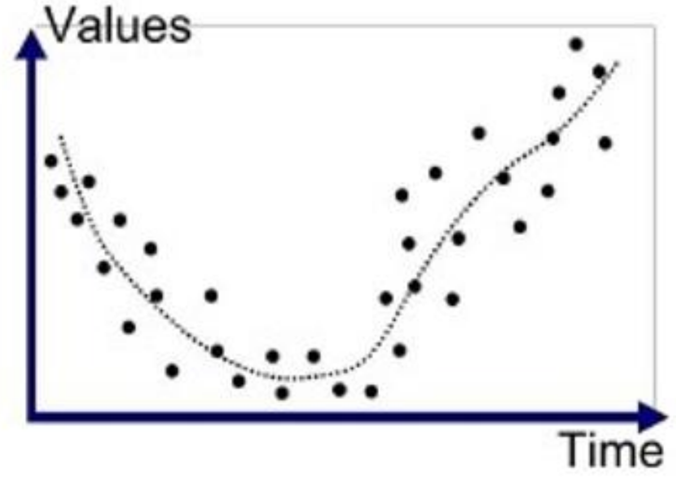


Underfitting and Overfitting in Machine Learning

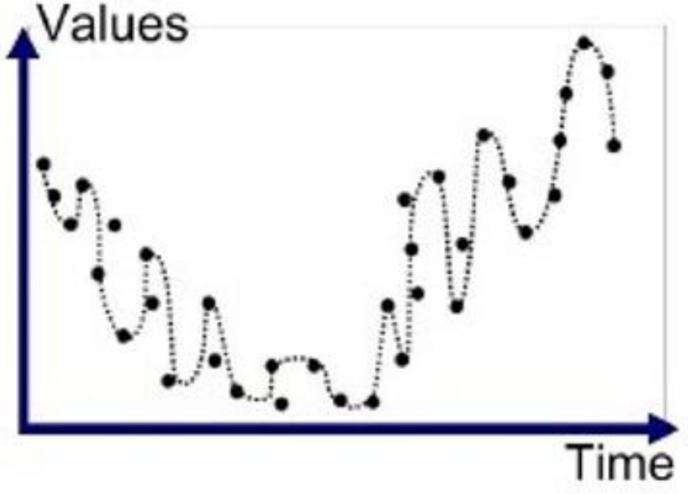
For an example: In a two-dimensional regression training fitting, we can observe that:



Underfitted



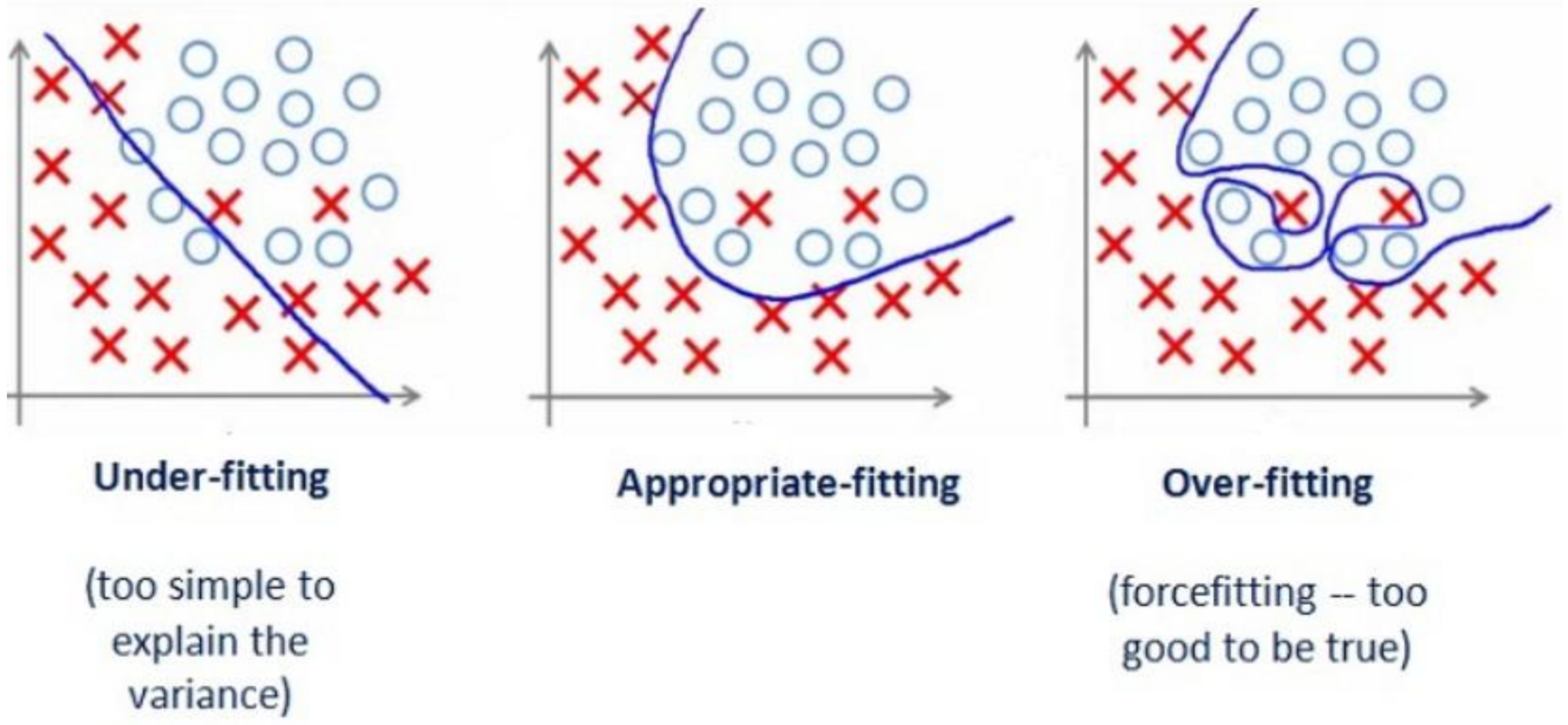
Good Fit/Robust



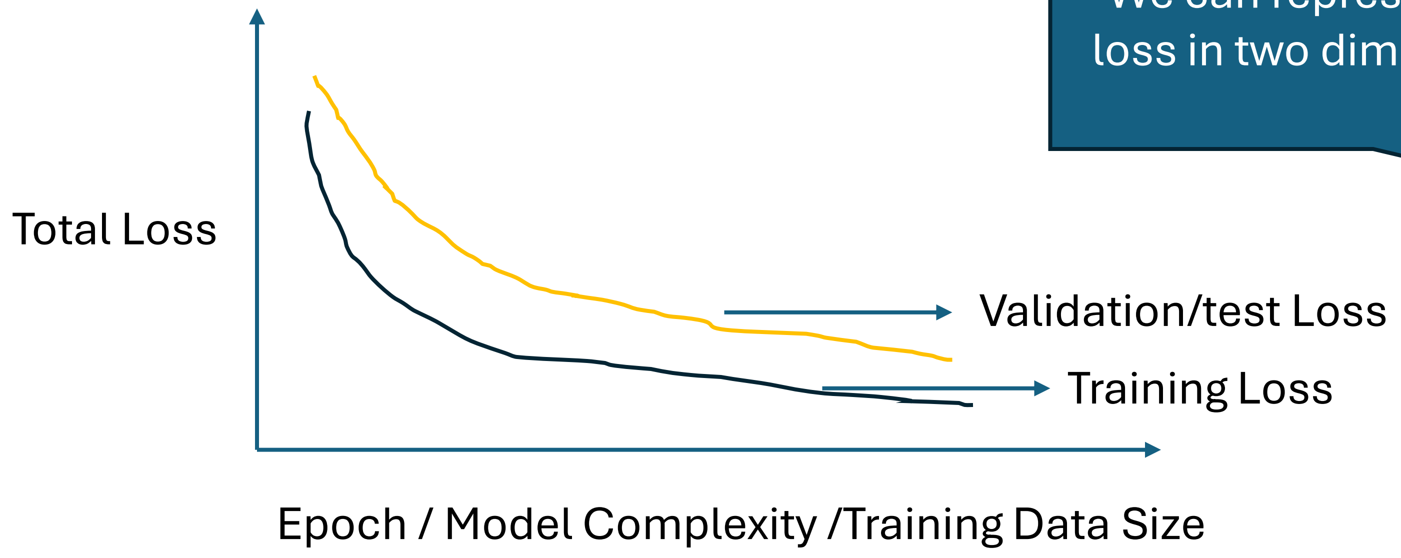
Overfitted

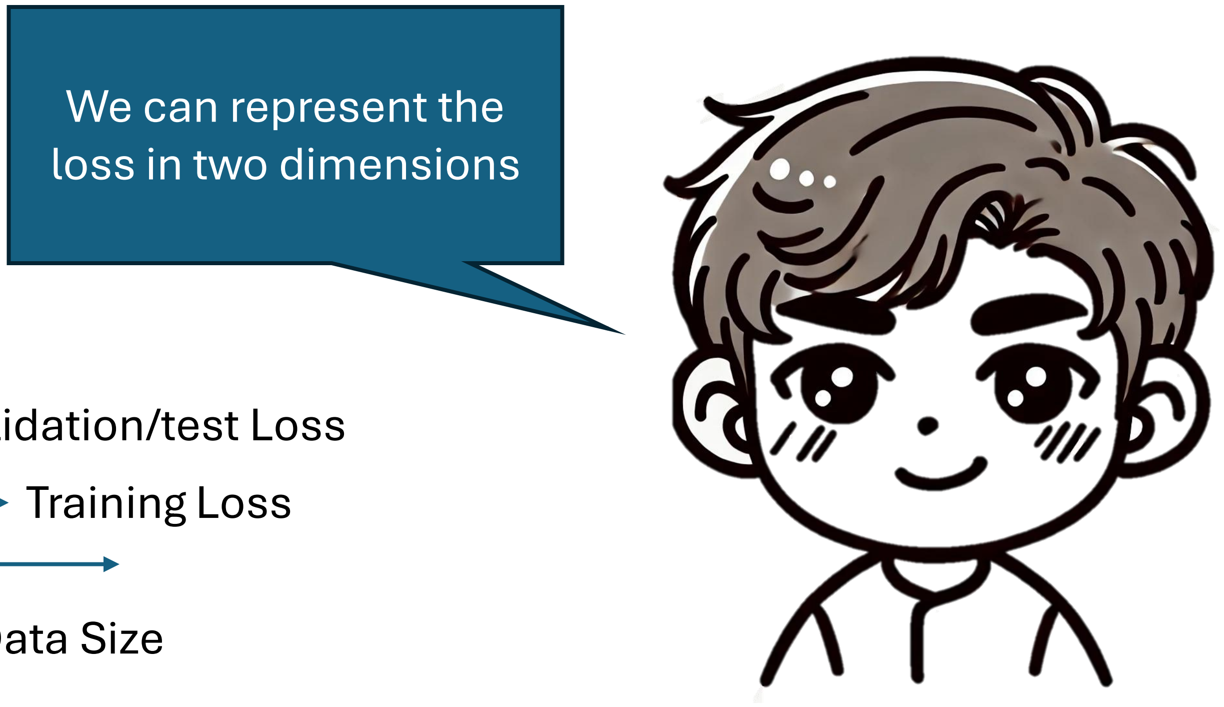
Underfitting and Overfitting in Machine Learning

In a two-dimensional classification training fitting, we can observe that

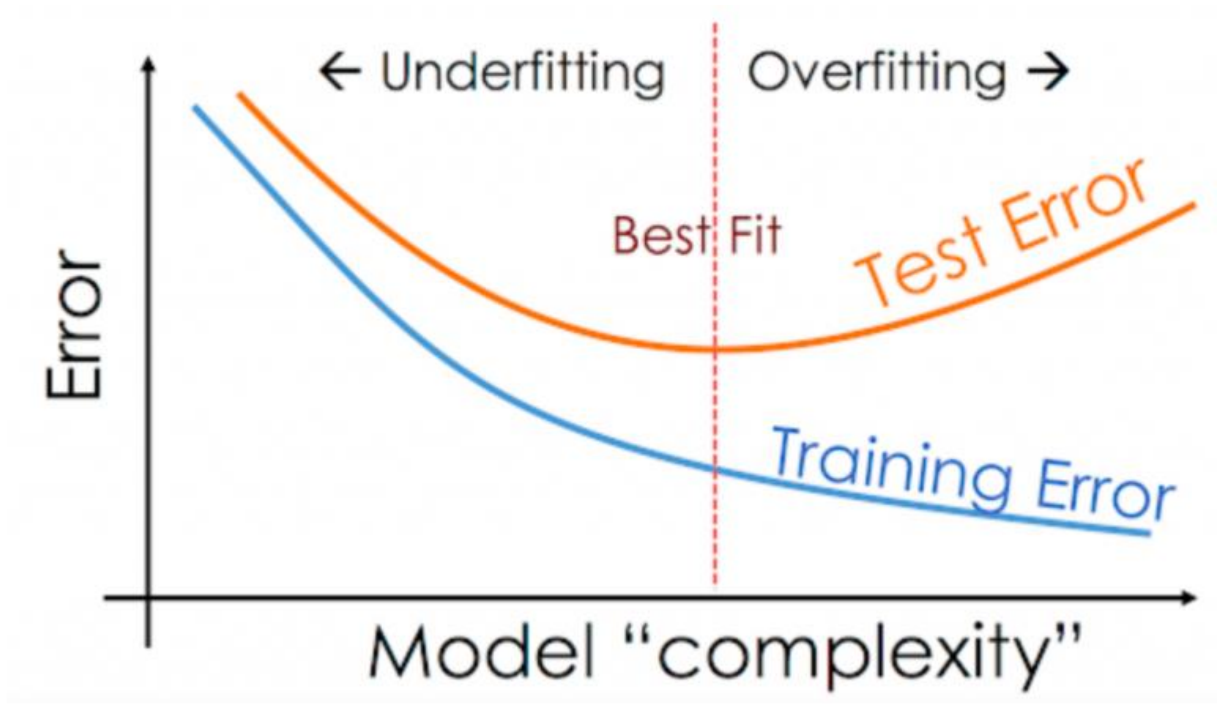

Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

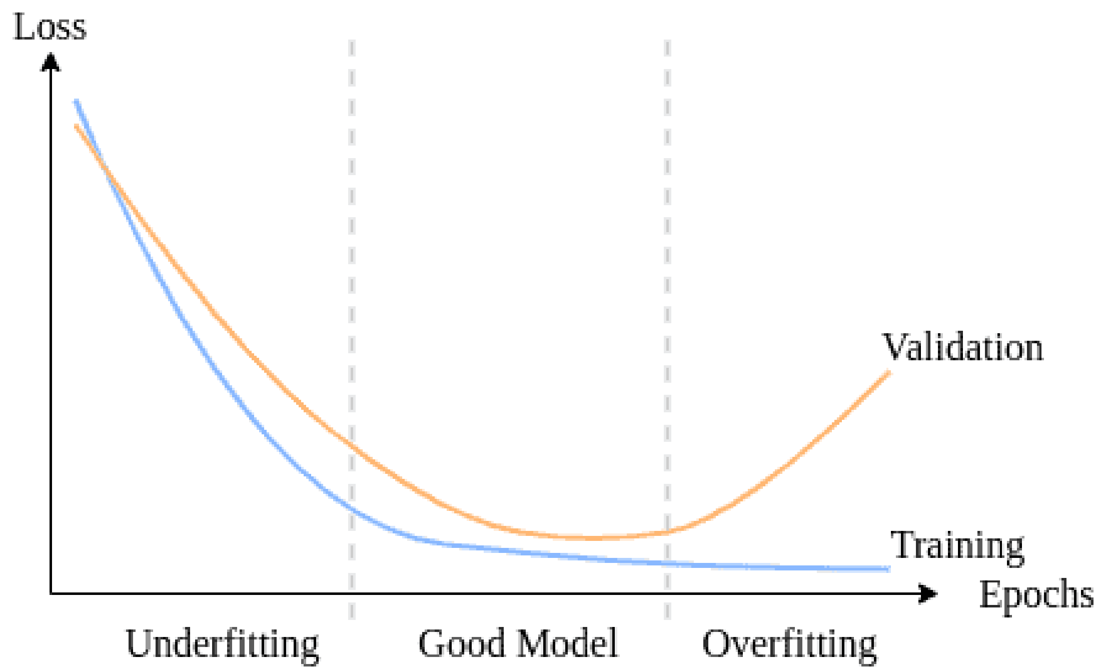
How can we determine this for models with more than two dimensions?





How can we determine this for models with more than two dimensions?





$\text{Bias}^2 + \text{Variance} + \text{Noise} = \text{Total Error}$

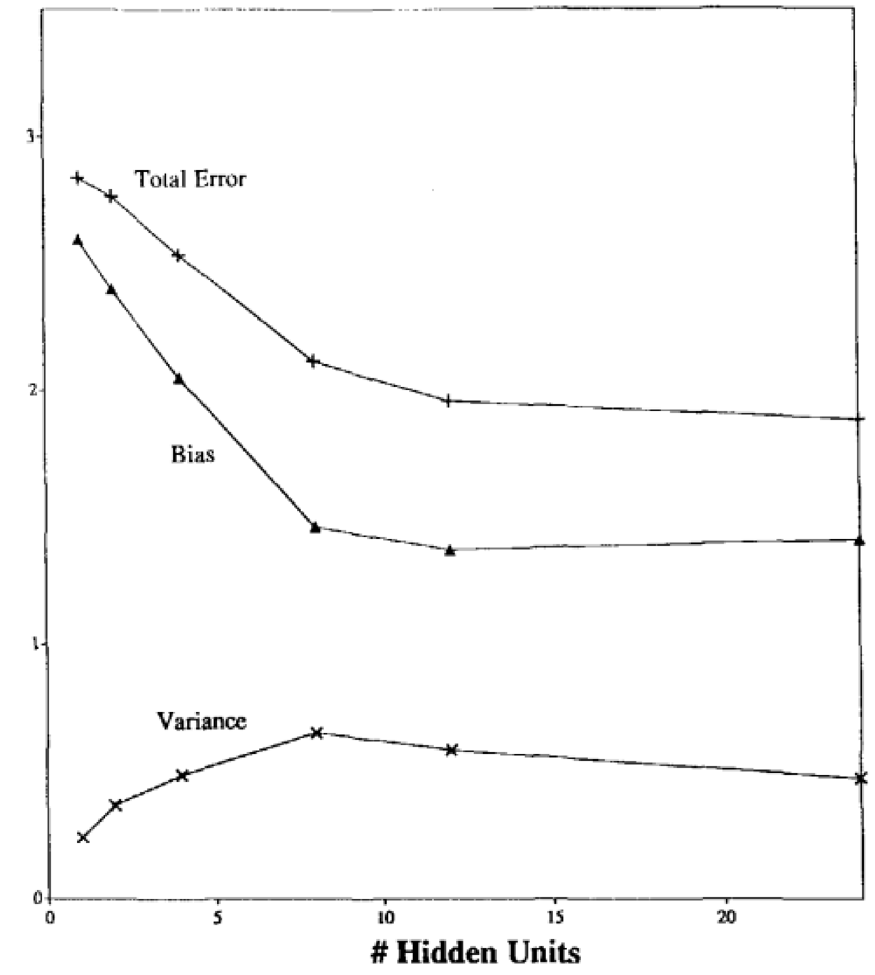
1992

Neural Networks and the Bias/Variance Dilemma

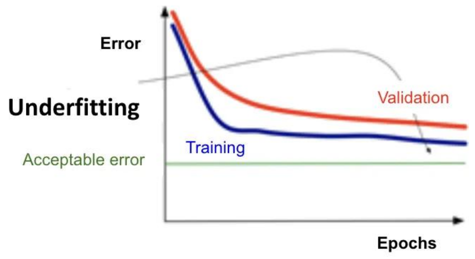
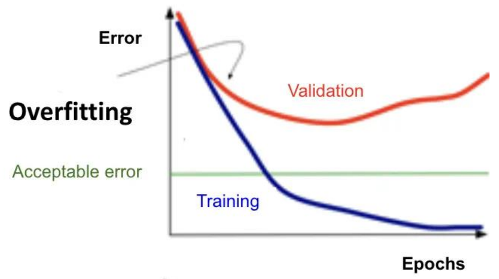
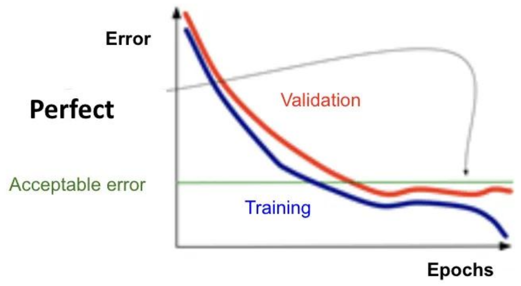
Stuart Geman
Division of Applied Mathematics,
Brown University, Providence, RI 02912 USA

Elie Bienenstock
René Doursat
ESPCI, 10 rue Vauquelin,
75005 Paris, France

Feedforward neural networks trained by error backpropagation are examples of nonparametric regression estimators. We present a tutorial on nonparametric inference and its relation to neural networks, and we use the statistical viewpoint to highlight strengths and weaknesses of neural models. We illustrate the main points with some recognition experiments involving artificial data as well as handwritten numerals. In way of conclusion, we suggest that current-generation feedforward neural networks are largely inadequate for difficult problems in machine perception and machine learning, regardless of parallel-versus-serial hardware or other implementation issues. Furthermore, we suggest that the fundamental challenges in neural modeling are about representation rather than learning per se. This last point is supported by additional experiments with handwritten numerals.



Overfitting vs Underfitting

Name	Definition	Characteristics	Bias / Variance
Underfitting	The model fails to capture the underlying structure of the data.	 High training and test errors	**High bias, low variance**
Overfitting	The model is too complex and learns even the noise in the data.	 Low training error but high validation error.	**Low bias, high variance**
Best Fit	The model appropriately captures the main trends without fitting noise.	 Both training and test errors are relatively low.	**Balanced bias–variance**

How Do We Address Overfitting and Underfitting

Overfitting

****Batch normalization,**
L1/ L2 regularization,
** **dropout,**
****Early Stopping**
Data augmentation

Underfitting

Add data, add layer,
check features

L2 Regularization

- Stabilizes gradient updates.
- Reduces overfitting by keeping weights small.
- Leads to smoother convergence. More stable.

$$L = Loss(\, y \,, \hat{y}\,)$$

$$L_{total} = Loss(y, \hat{y}) + \lambda_i \sum w_i^2$$

Penalty

It is a coefficient. If we set it larger, we get a stronger penalty

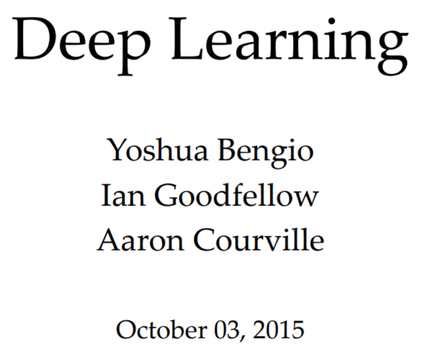
When we do gradient decent!

$$w := w - \eta\left(\frac{\partial L}{\partial w} + 2\lambda w\right) = (1 - 2\eta\lambda)w - \eta\frac{\partial L}{\partial w}$$

$\longrightarrow$

If $w > 0$, it shrinks toward 0 due to the term $(1 - 2\eta\lambda)$.

If $w < 0$, it grows toward 0 (less negative) for the same reason.



Deep Learning

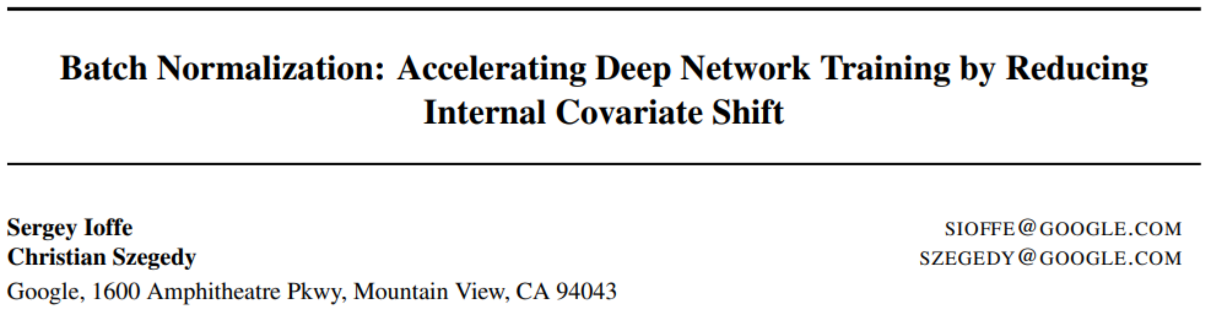
Yoshua Bengio
Ian Goodfellow
Aaron Courville

October 03, 2015

Check chapter 7.1, it must consider $\eta\lambda$ together,

$$0 < \eta < 1$$
$$0 < \lambda < 1$$

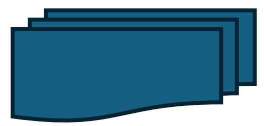
Batch Normalization



Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe SIOFFE@GOOGLE.COM
Christian Szegedy SZEGEDY@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

Core idea:



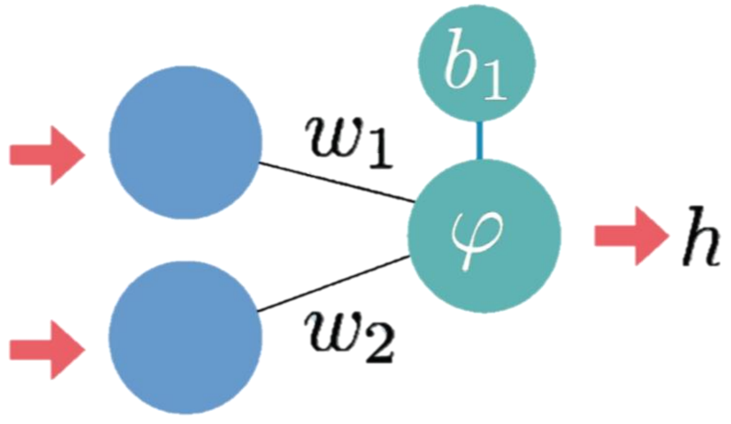
A batch data (Not so small): e.g. 32, 64, 100.....

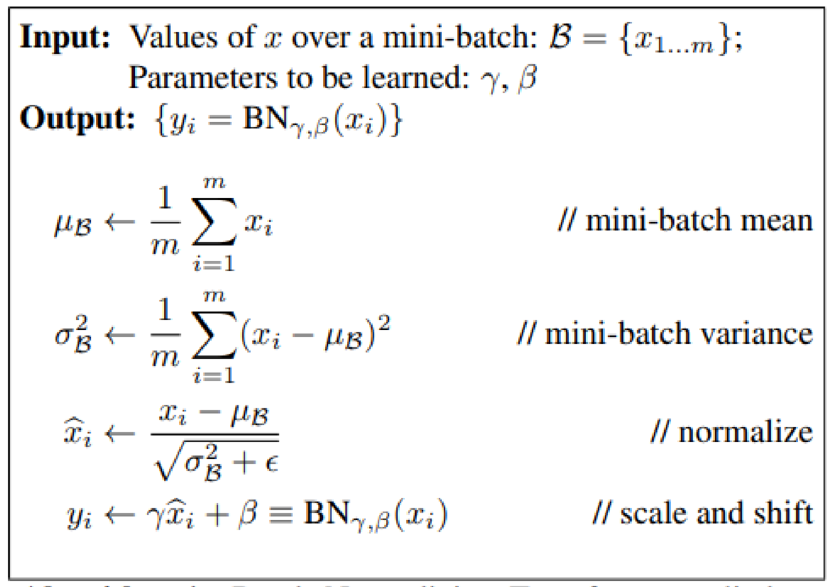
Each data with:

x_1

x_2

$$\hat{x} = \frac{x - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}}$$





Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

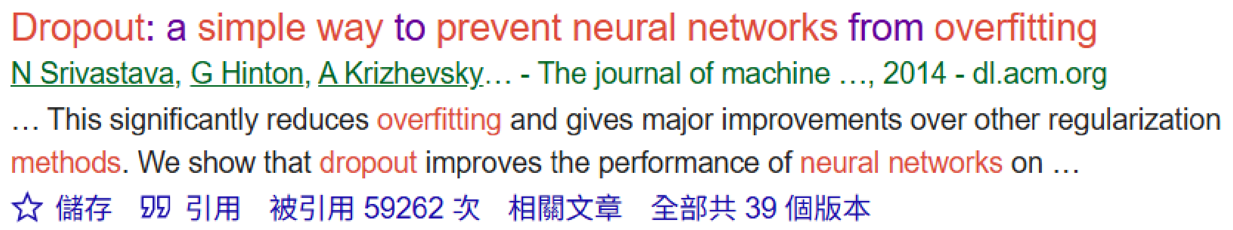
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Effects of Batch Normalization
- Allows higher learning rates
- Reduces sensitivity to initialization
- Accelerates convergence
- Acts as a regularizer (can reduce or replace Dropout)

Dropout

The guy comes again.....



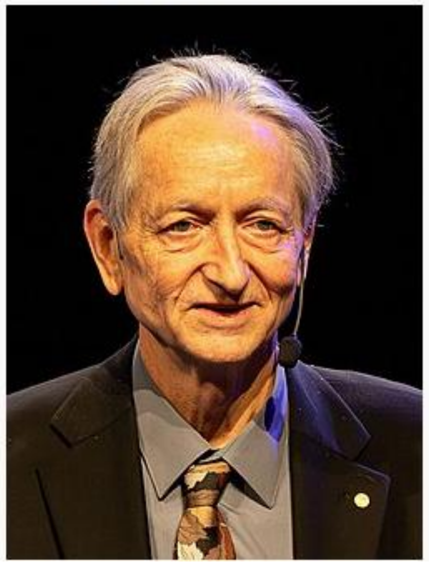
Dropout: a simple way to prevent neural networks from overfitting

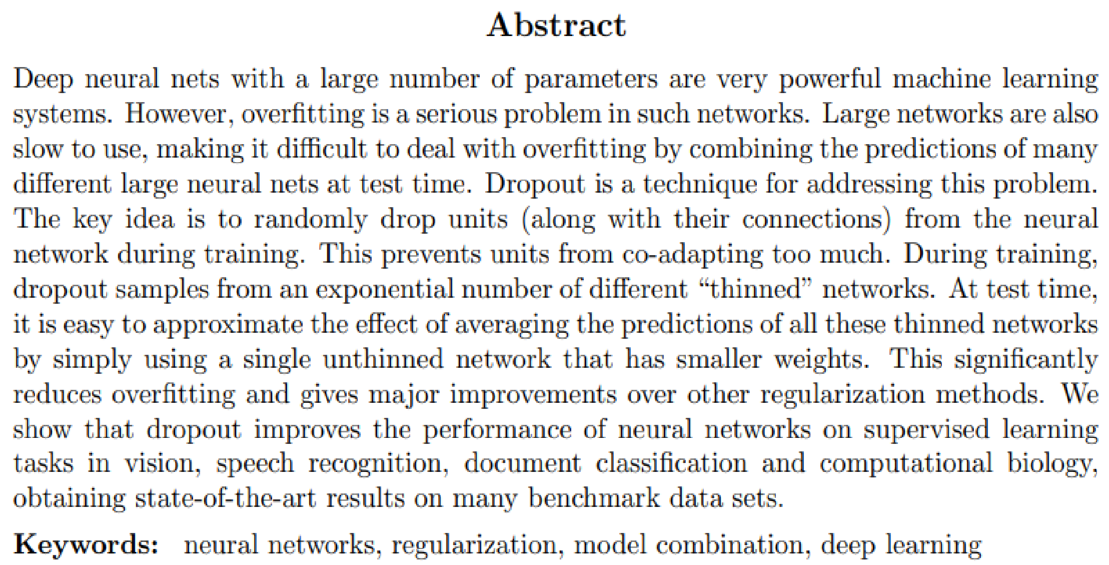
N Srivastava, G Hinton, A Krizhevsky… - The journal of machine …, 2014 - dl.acm.org

… This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on …

☆ 儲存 99 引用 被引用 59262 次 相關文章 全部共 39 個版本

Geoffrey Hinton

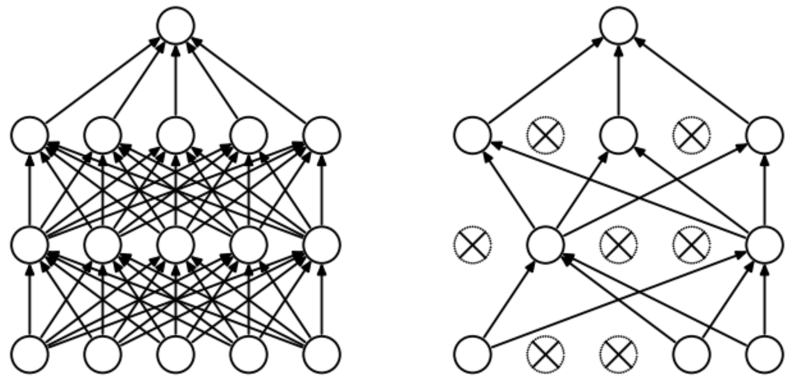




Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning



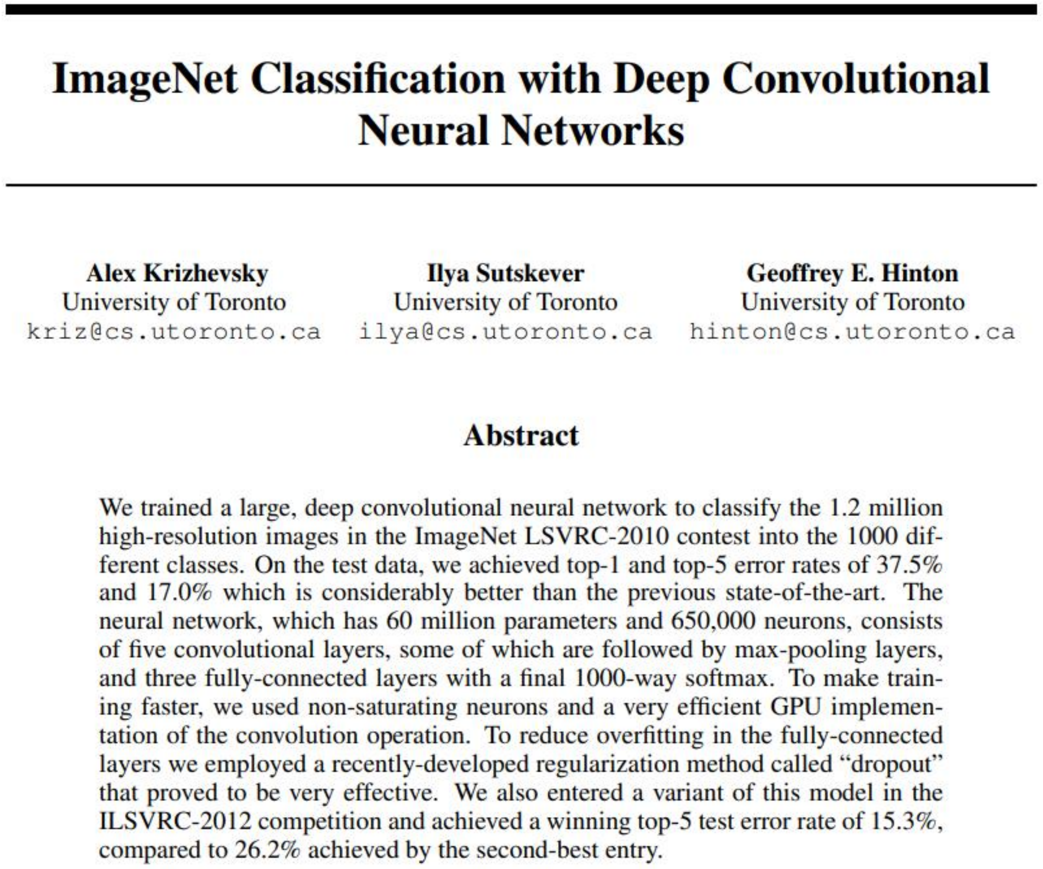
(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

This is different from **Monte Carlo dropout !**

Data augmentation

The guy comes again and again........



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

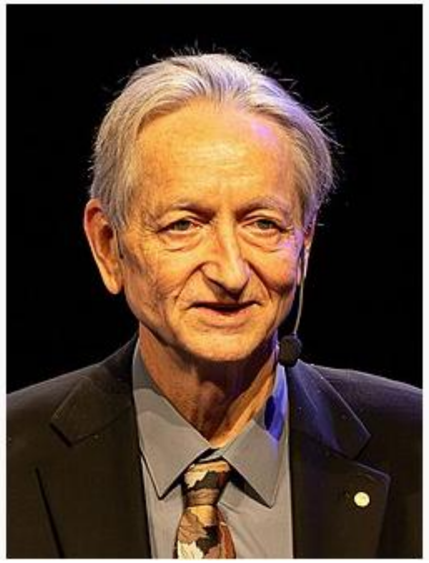
Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Alex Krizhevsky

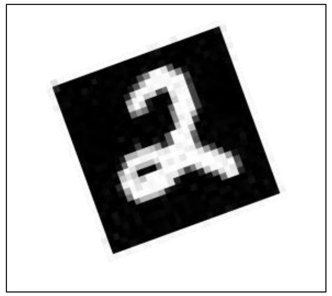
Geoffrey Hinton

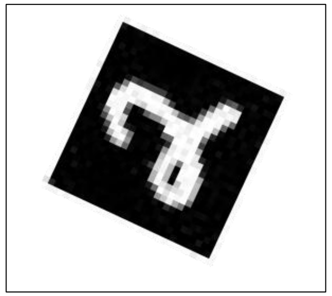


Ilya Sutskever (OpenAI co-founder)

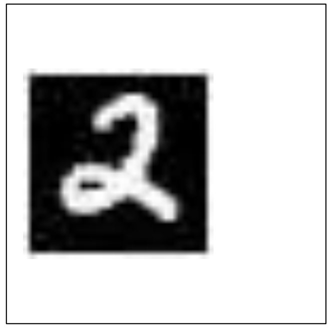
label = 2

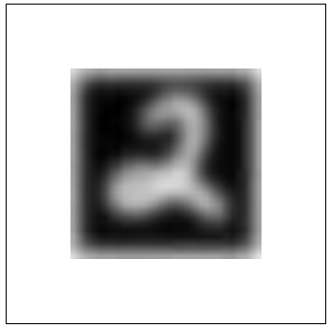
[given training data]
We exploit the fact that certain transformations to the image do not change the label of the image.


rotated by 20°


rotated by 65°

shifted vertically


shifted horizontally


blurred

changed some pixels

label = 2

[augmented data = created using some knowledge of the task]

- Typically, More data = better learning
- Works well for image classification / object recognition tasks.

  Also shown to work well for speech
- For some tasks it may not be clear how to generate such data such as NLP

Hands-on Exercise

- Use the same Colab file as last week
- Try to add dropout, batch normalization, L2 Regularization to improve your model

Model Calibration

What is model calibration?

Problem:

- ML models are often over-confident or under-confident.
- Softmax outputs are not guaranteed to be **real probabilities**.

Definition:

- A model is well-calibrated (perfect calibration) as:

$$\mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) = p, \quad \forall p \in [0, 1]$$

True correctness probability

predicted label

predicted confidence for a sample

predicted label for a sample

predicted confidence

Goal:

Make predicted confidence match true likelihood→ If model predicts 0.8 probability, about 80% should be correct.

Reference: Guo et al., "On Calibration of Modern NeuralNetworks", ICML 2017

Model Calibration

What is model calibration?

Motivation:

- Enables trustworthy decision-making (e.g., risk assessment, medical AI).
- Improves interpretability of model confidence

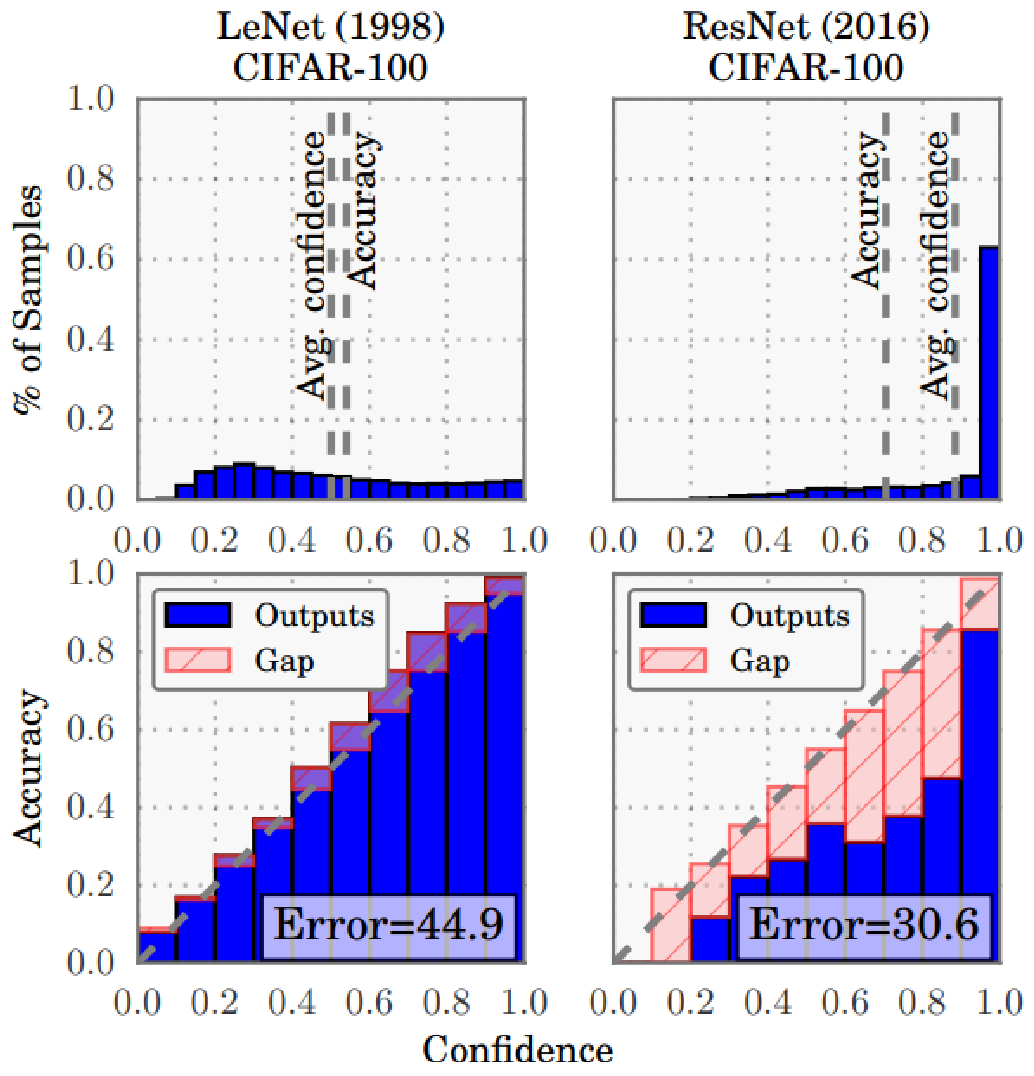


Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

Reference: Guo et al., "On Calibration of Modern Neural Networks", ICML 2017

Post-hoc Calibration Techniques

Platt Scaling (Platt, 1999): Learn sigmoid mapping

$$P^{\text{cal}} = \frac{1}{1 + e^{(Az+B)}}$$

- Simple, effective for binary tasks

Temperature Scaling (Guo et al., 2017):

$$P_i^{(T)} = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

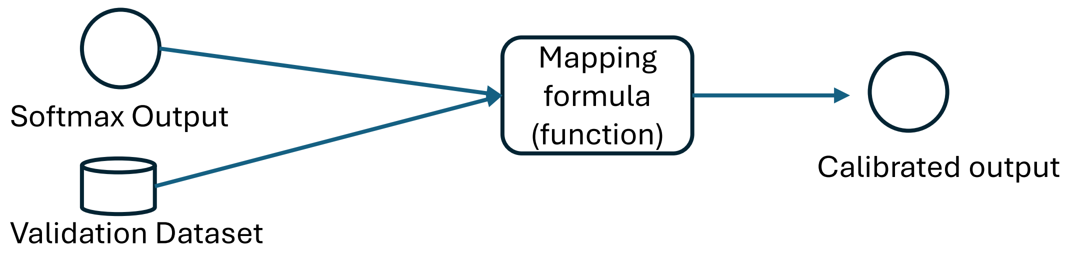
- Most common for deep nets, preserves ranking

Histogram Calibration

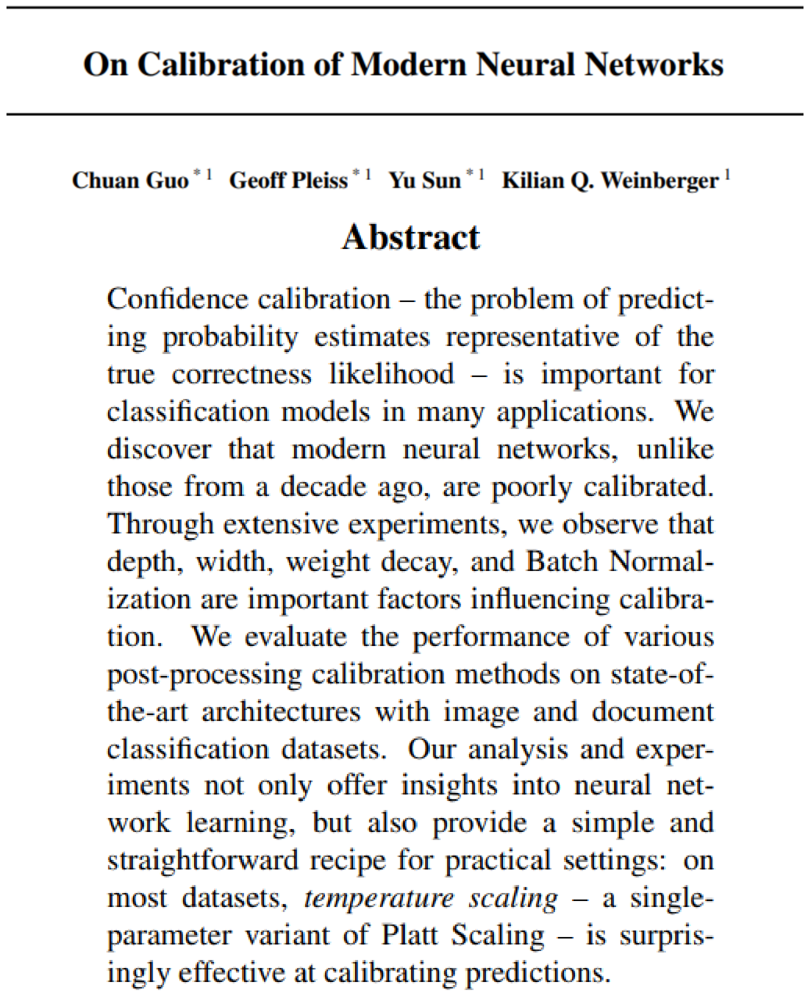
- Fit distribution over confidence bins

Core idea:

Calibration adjusts confidence
→ approximates true posterior probability.



ICML 2017, 8000 up citations



On Calibration of Modern Neural Networks

Chuan Guo [*1] Geoff Pleiss [*1] Yu Sun [*1] Kilian Q. Weinberger [1]

Abstract

Confidence calibration – the problem of predicting probability estimates representative of the true correctness likelihood – is important for classification models in many applications. We discover that modern neural networks, unlike those from a decade ago, are poorly calibrated. Through extensive experiments, we observe that depth, width, weight decay, and Batch Normalization are important factors influencing calibration. We evaluate the performance of various post-processing calibration methods on state-of-the-art architectures with image and document classification datasets. Our analysis and experiments not only offer insights into neural network learning, but also provide a simple and straightforward recipe for practical settings: on most datasets, *temperature scaling* – a single-parameter variant of Platt Scaling – is surprisingly effective at calibrating predictions.

A simple example

$z = [2, 1, 0.1]$

$\rightarrow e^2 = 7.39, e^1 = 2.72, e^{0.1} = 1.11$

$\rightarrow Sum = 11.22$

$\rightarrow P = [0.659, 0.243, 0.099]$

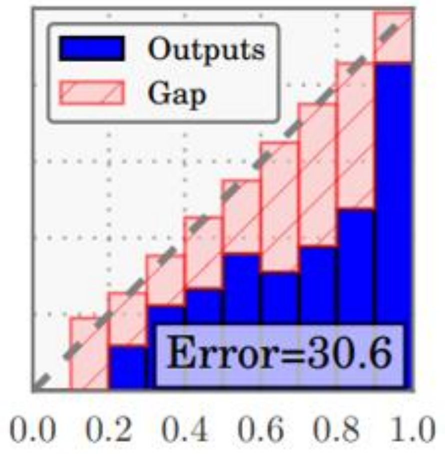
The model predicts class A with a confidence of 0.659

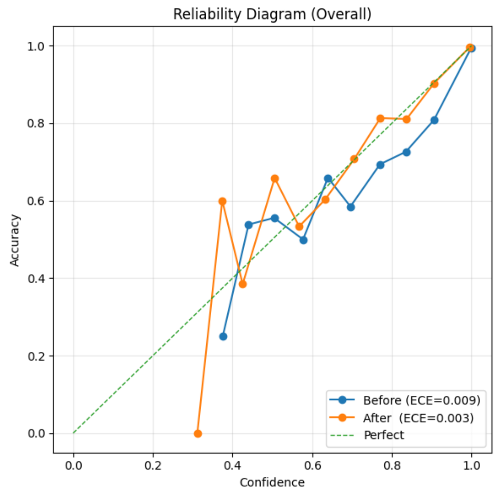
Formula:

$\rightarrow$ $$P_i^{(T)} = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

$\rightarrow$ Use the validation set to find the temperature that best minimizes cross-entropy.

Evaluating Calibration





Reliability Diagram:

- Plot predicted confidence vs. empirical accuracy
- Perfect calibration → points on diagonal

(Show: under-confident below line, over-confident above line)

Expected Calibration Error (ECE) (Naeini et al., 2015):

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \right|,$$

M: The number of bins (usually 10 or 15)

- → Smaller is better.

Summary:

- Before calibration: outputs are raw confidences (not true probabilities).
- After calibration: outputs are probabilistically meaningful.
- Calibrated models support risk-aware decision-making.

References: Guo et al., ICML 2017

Hands-on Exercise

- Use the same Colab file as last week