Workshop on Hands-on Deep Learning Coding and Code Management

Organized by

Center for Computational & Data Sciences, IUB

Day 1

Who we are?

Dr. AKM Mahbubur Rahman

Associate Professor, Director Data Science wing

Research Assistants

Md Fahim	Moshiur
Mir Sazzat Hossain	Iftee
Jahir Sadik Monon	Fahim Ahmed
Armun Alam	Dehan

Why we are here?

- ✓ Get introduced to deep learning programming
- ✓ Practice programming
- ✓ Develop deep learning models
- ✓ Training and fine tuning deep learning models
- ✓ Hands on experiment design and result analysis
- ✓ Guidelines for standard coding practice for deep learning

Preferred skills

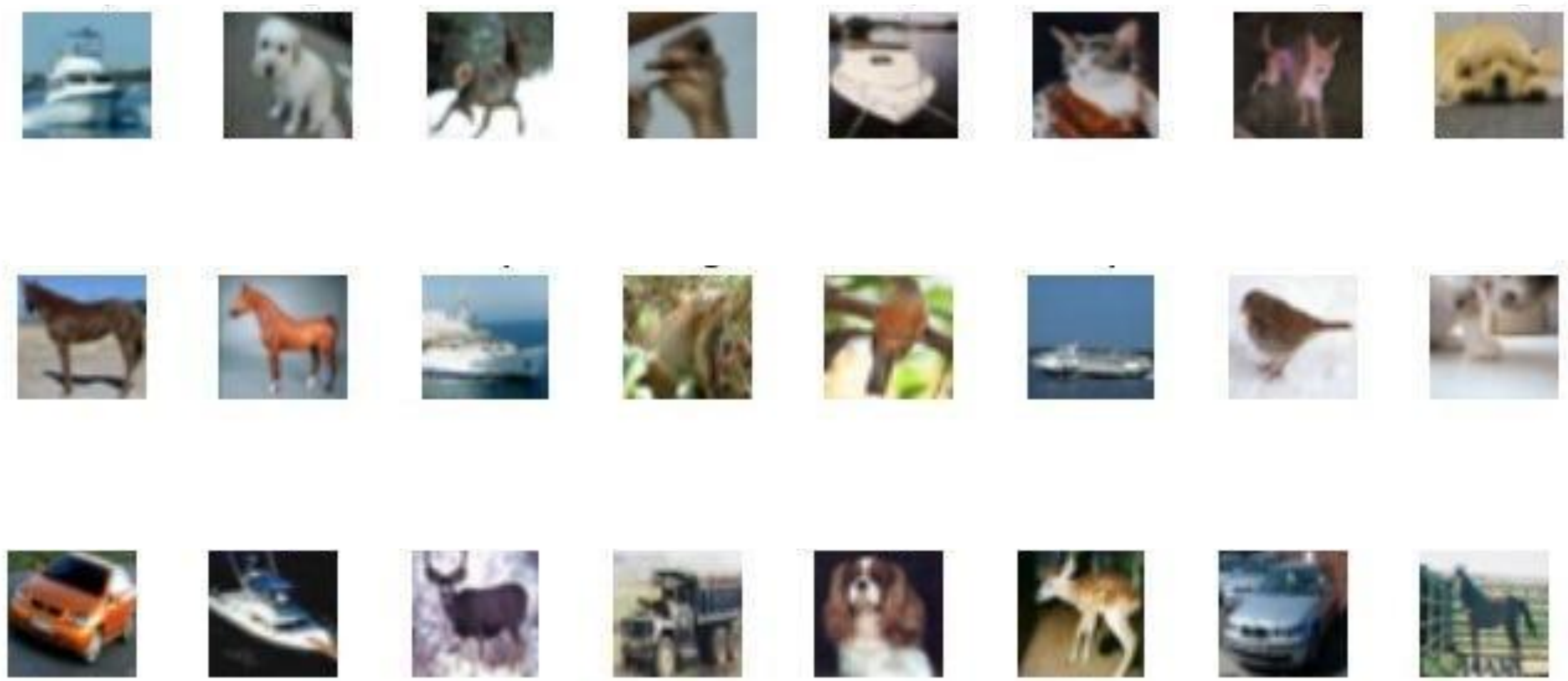
- ✓ Python basics with numpy
- ✓ Finished Numerical Methods course
- ✓ Linear Algebra with vector notations
- ✓ Matplotlib, pyplot for visualization
- ✓ AI, ML, Data Mining courses

Day 1

- ✓ Image Classification Task
    - With a custom CNN, CIFAR10
    - Use of pretrained VGG 16
- ✓ Babysitting your CNNs
- ✓ Natural Language Processing Tasks (Emotion recognition from sentences) using
    - LSTM
    - Transformer (BERT)

Visual Recognition

Image Classification



Disclaimer: *Some slides are modified and adopted from CSE231n (CS231n: Deep Learning for Computer Vision) , Stanford University*

Visual Recognition

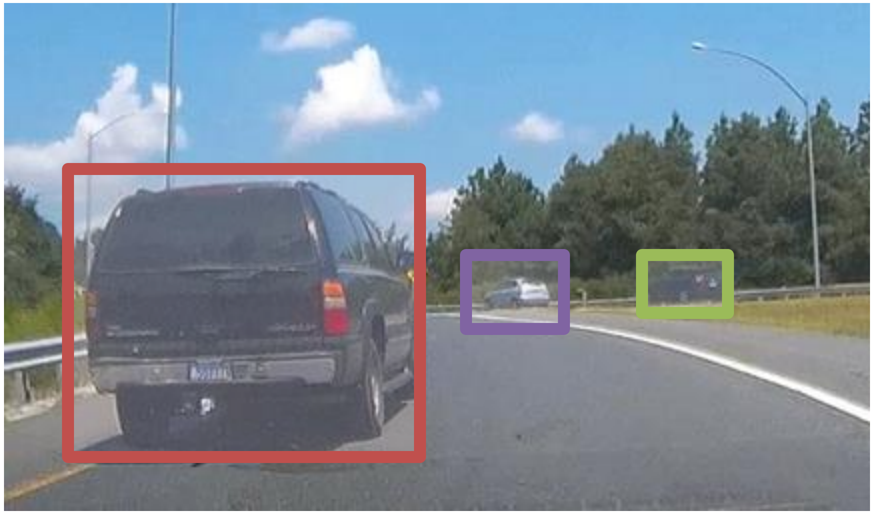
Image Classification

Visual Recognition

Image Classification

Visual Recognition

Image Classification



- **Object detection**
- **Action classification**
- **Image captioning**

Person

Hammer

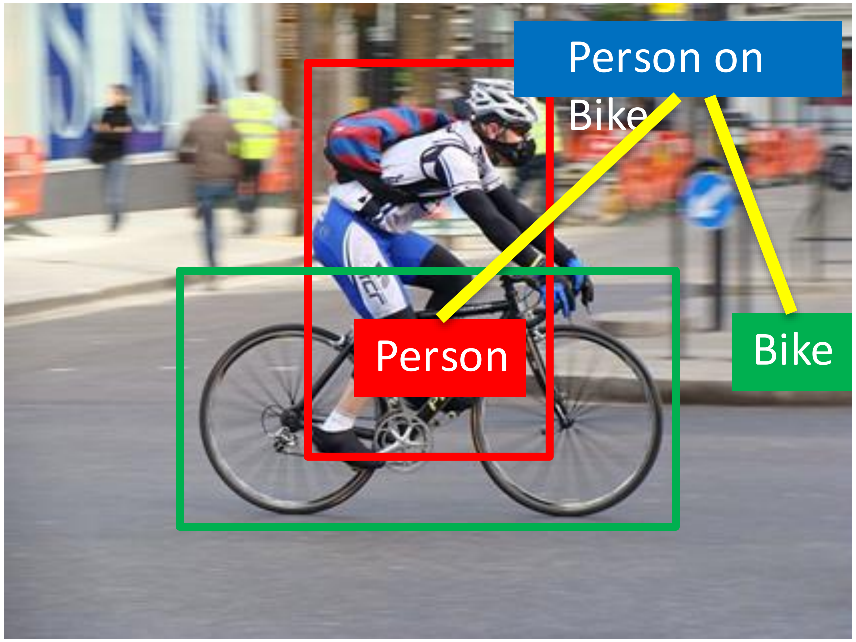


Image Classification pipeline

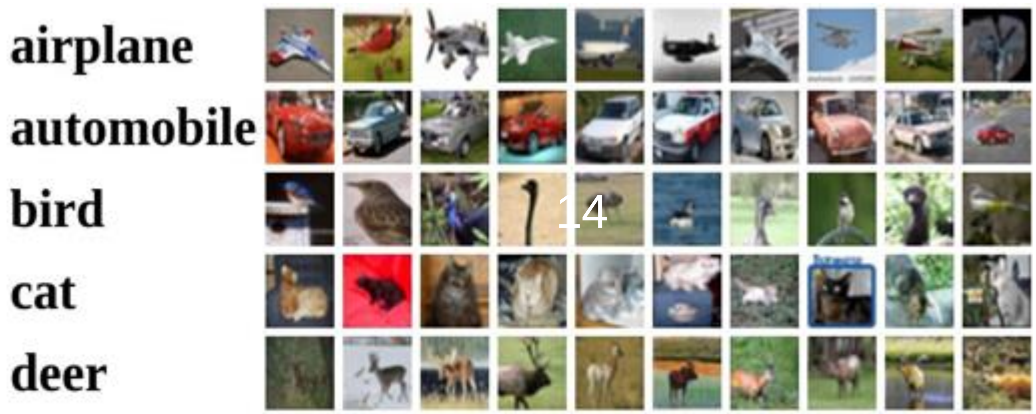
Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```python
def train(images, labels):
  # Machine learning!
  return model
```

```python
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set



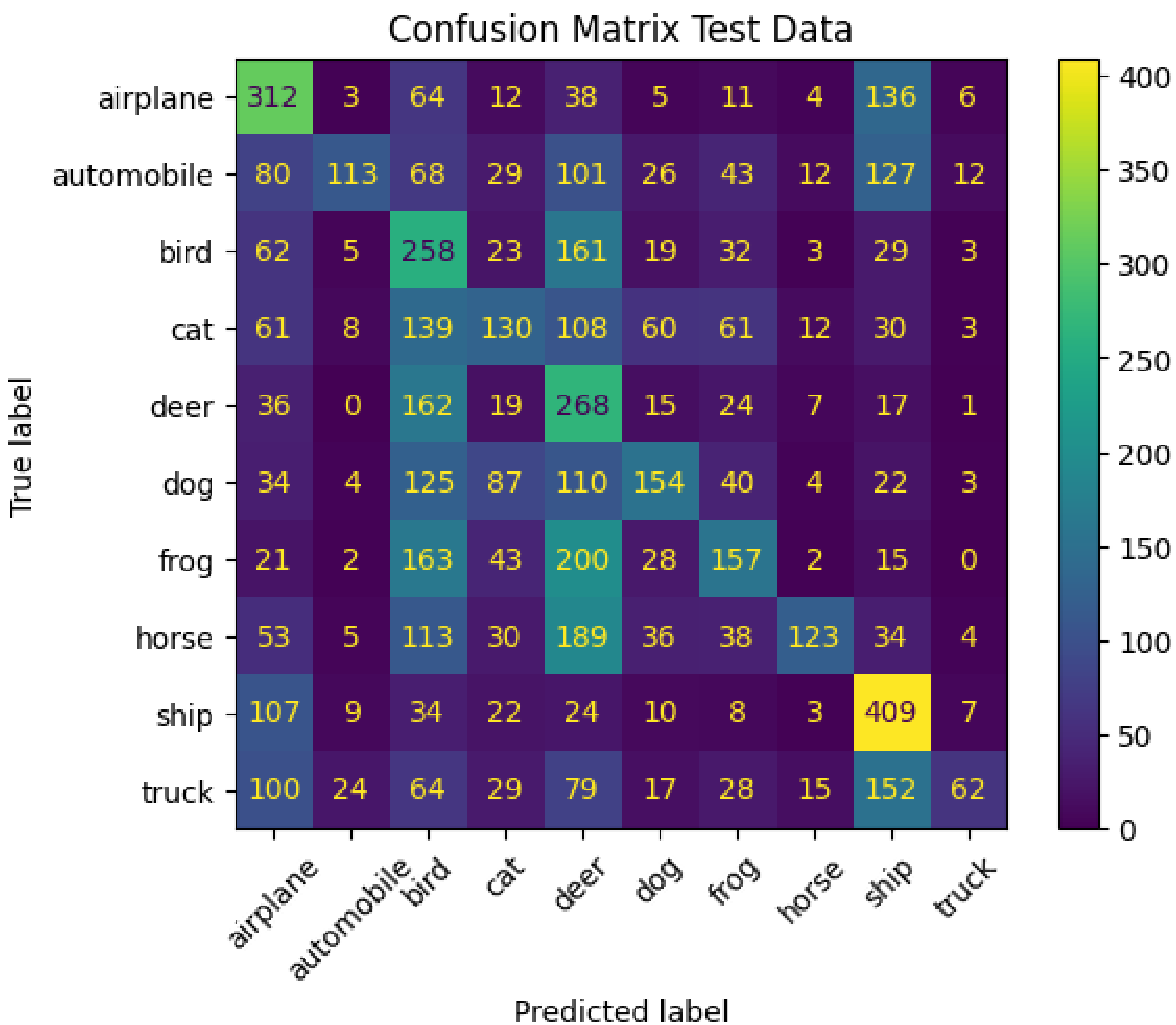
Example Dataset: **CIFAR10**

10 classes
50,000 training images
10,000 testing images

Test images and nearest neighbors

Alex Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Technical Report, 2009.

Evaluation in Test images



Evaluation in Test images

	precision	recall	f1-score	support
airplane	0.36	0.53	0.43	591
automobile	0.65	0.18	0.29	611
bird	0.22	0.43	0.29	595
cat	0.31	0.21	0.25	612
deer	0.21	0.49	0.29	549
dog	0.42	0.26	0.32	583
frog	0.36	0.25	0.29	631
horse	0.66	0.20	0.30	625
ship	0.42	0.65	0.51	633
truck	0.61	0.11	0.18	570
accuracy			0.33	6000
macro avg	0.42	0.33	0.32	6000
weighted avg	0.42	0.33	0.32	6000

Day 1, Session 1

Convolutional Neural Network

Big Picture

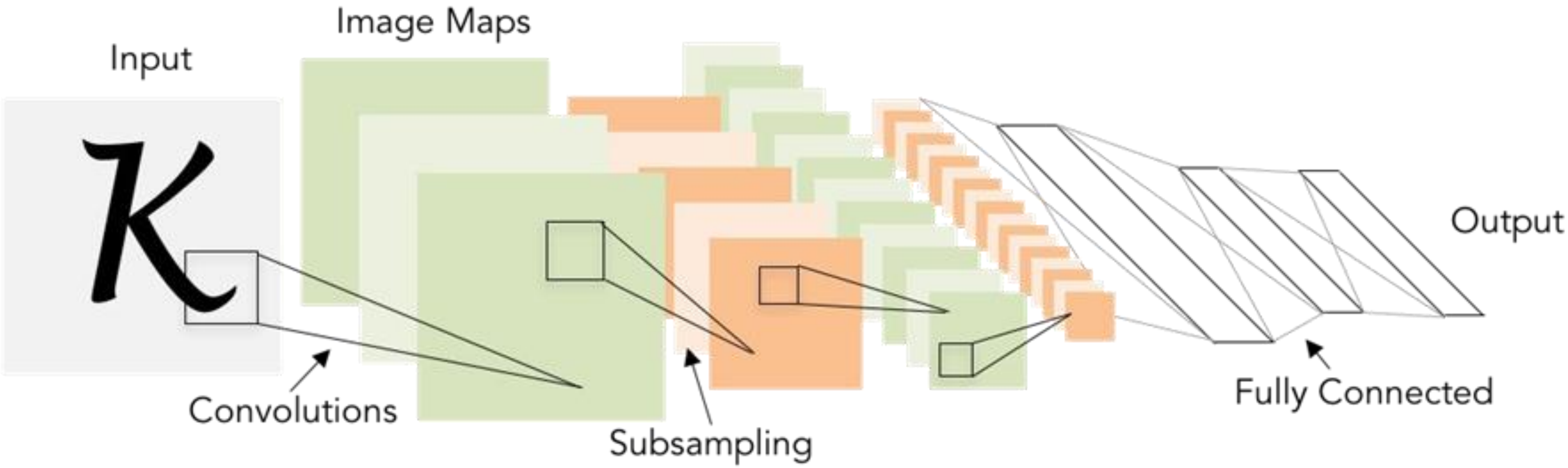
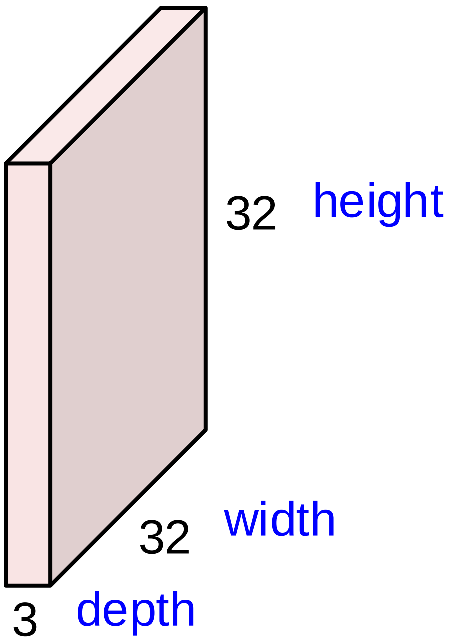


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

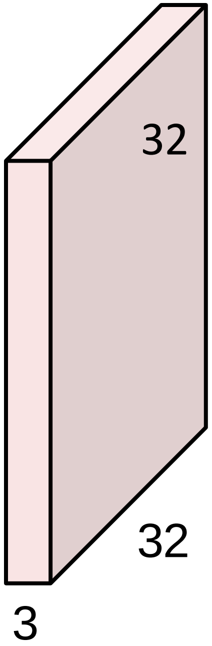
Convolution Layer

32x32x3 image -> preserve spatial structure

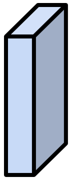


Convolution Layer

32x32x3 image



5x5x3 filter

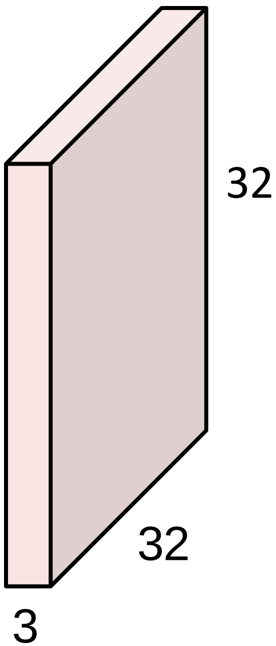


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

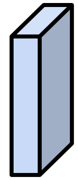
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

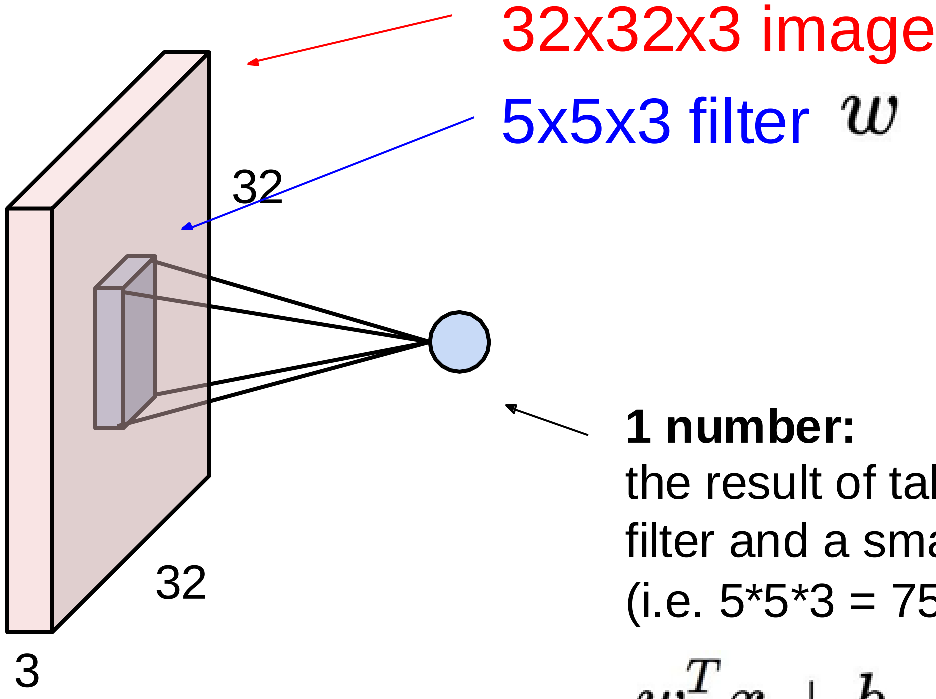


5x5x3 filter



Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer

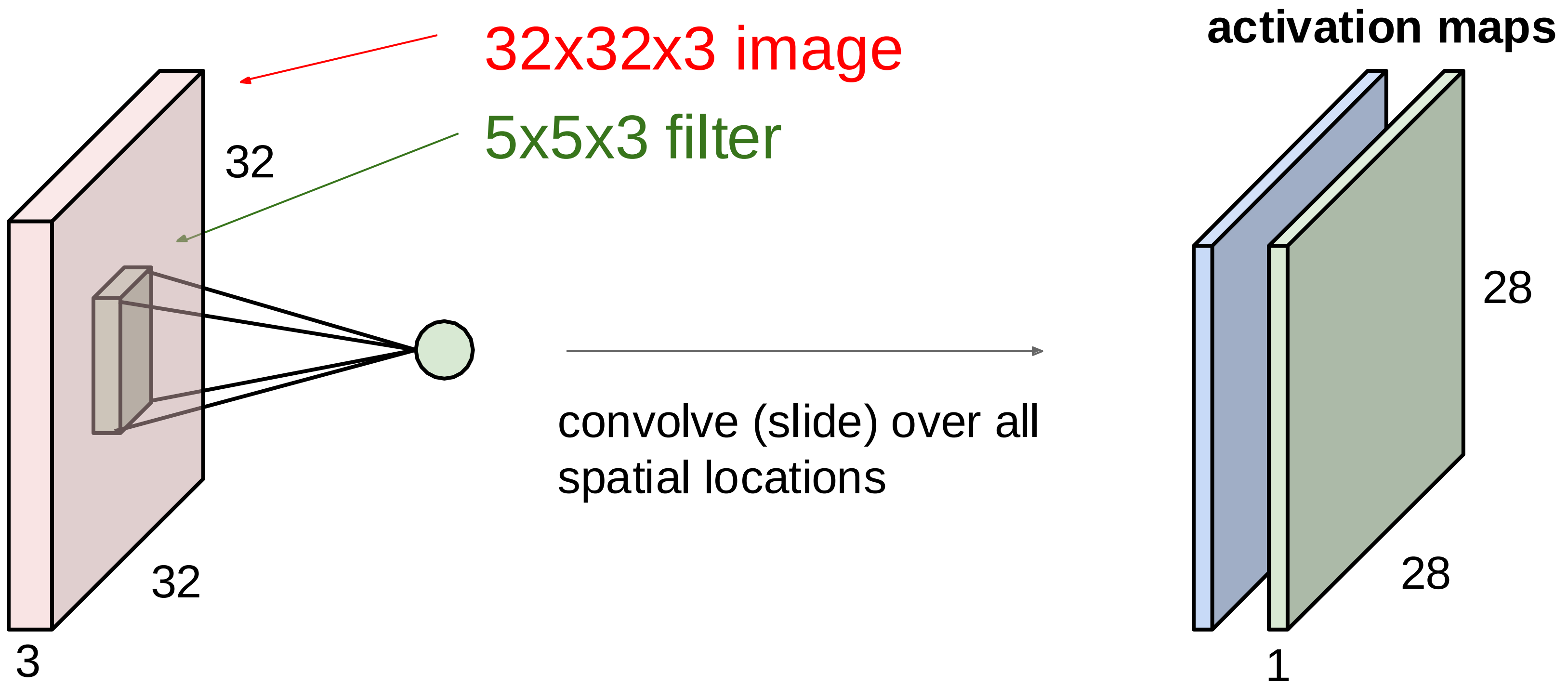


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

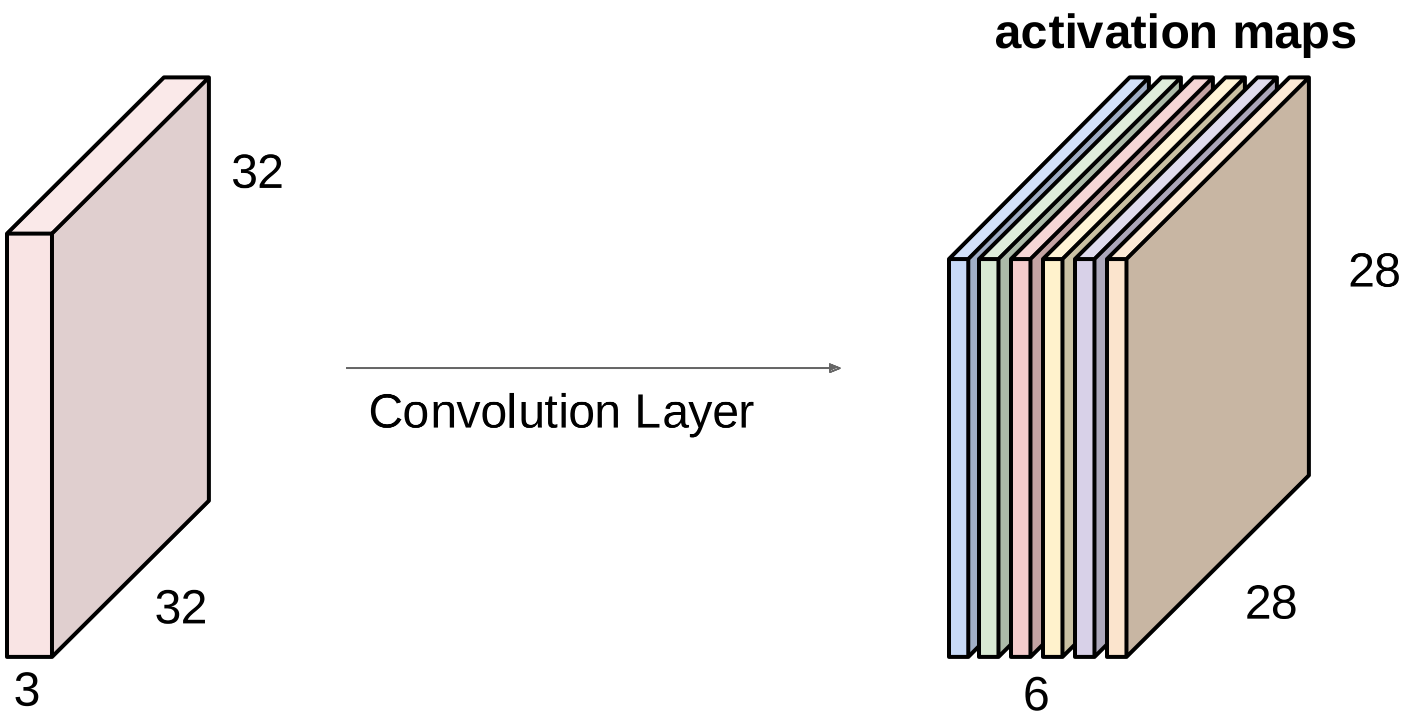
consider a second, green filter



Convolution Demo

http://cs231n.github.io/convolutional-networks/#conv

We have six filters



We stack these up to get a “new image” of size 28x28x6!

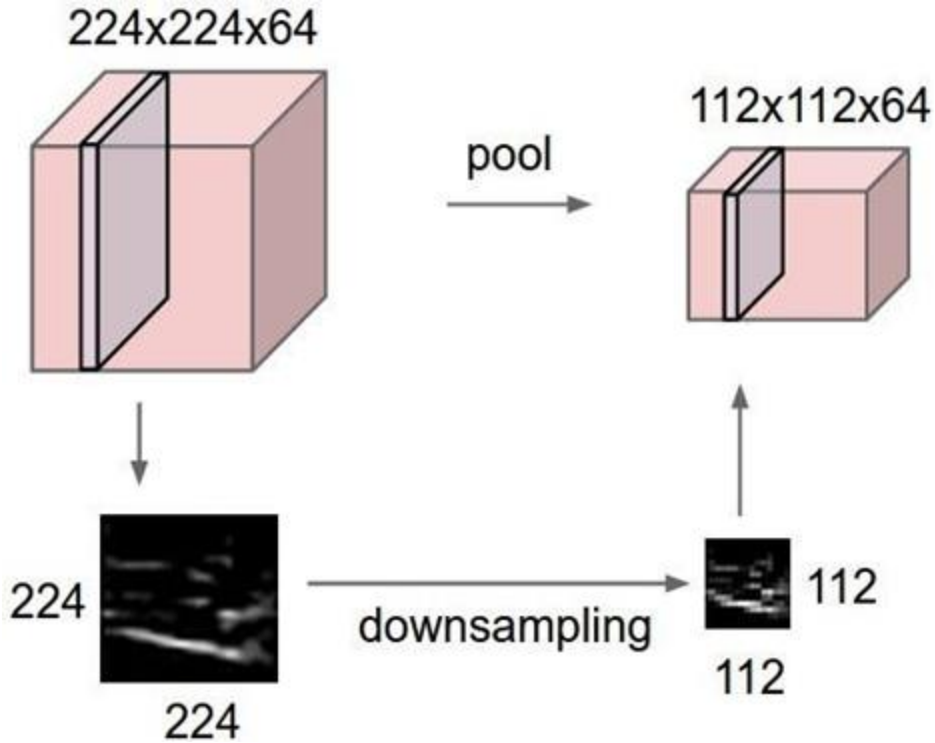
Summary of convolutional layer

Summary. To summarize, the Conv Layer:

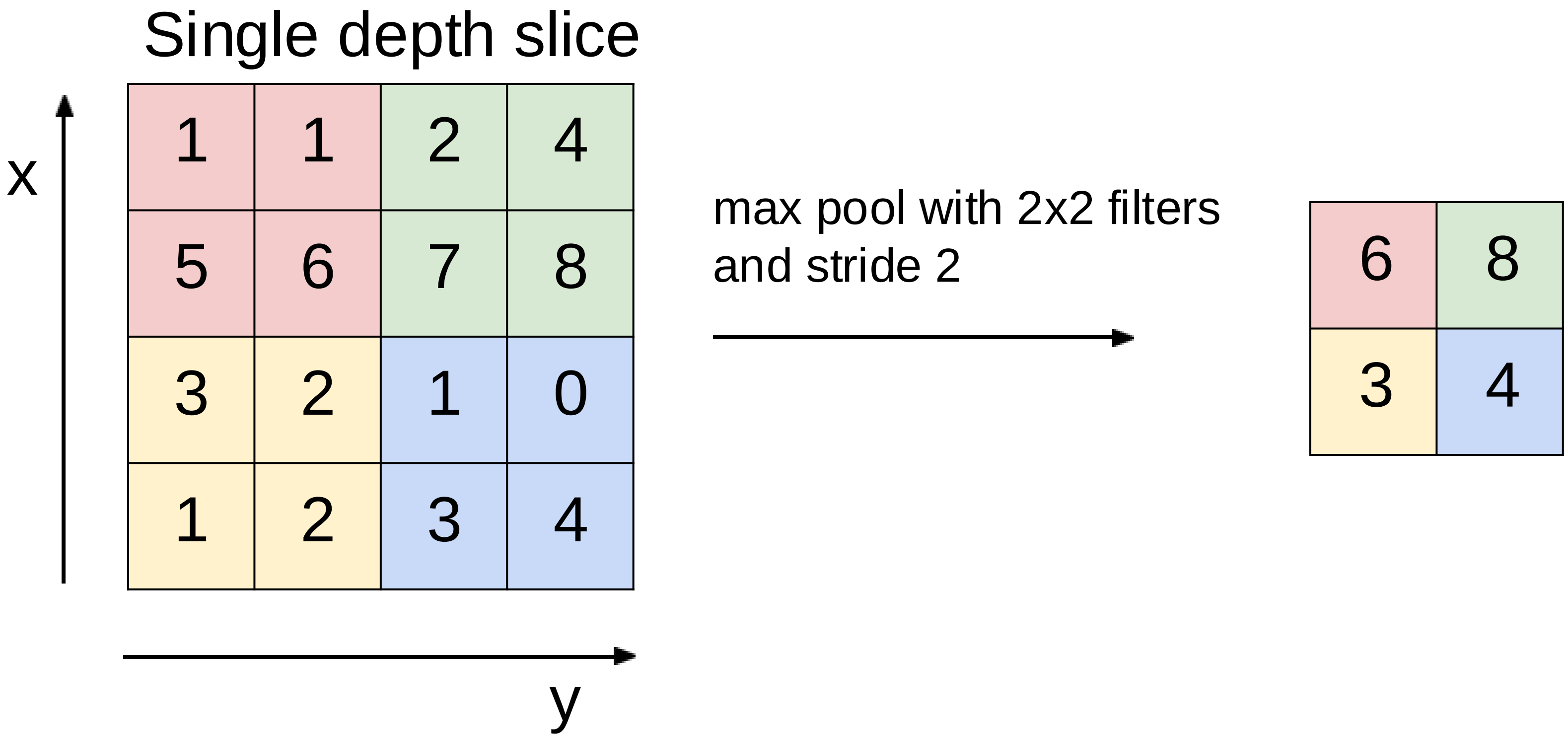
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Sub-sampling/Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Max Pooling



Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

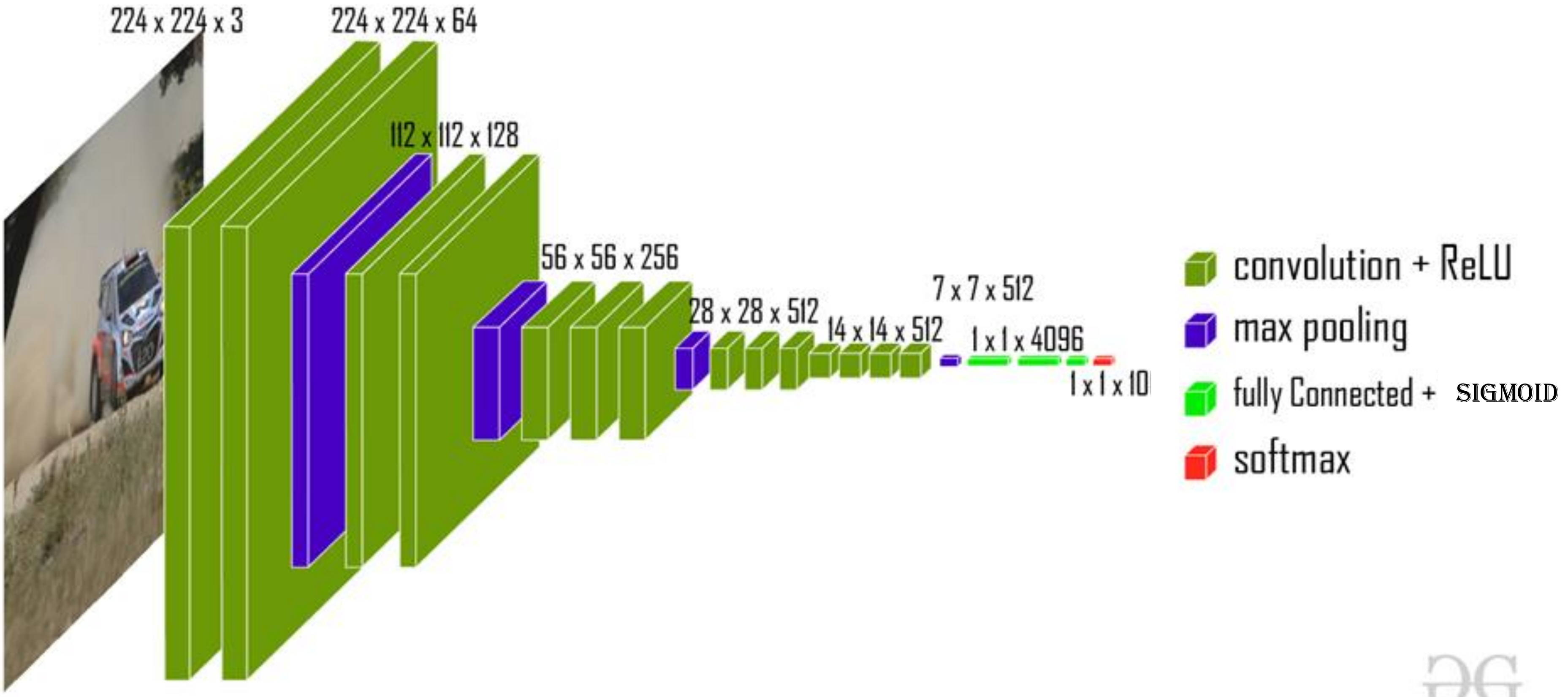
Summary of maxpool layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common settings:

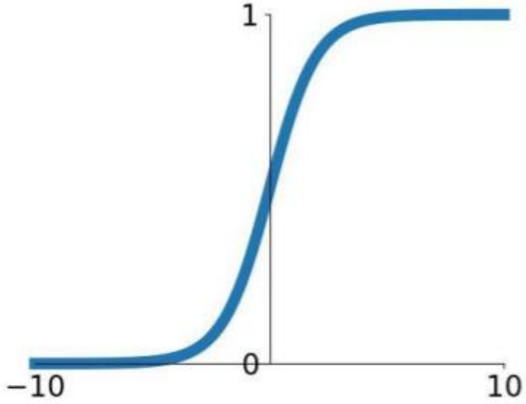
F = 2, S = 2
F = 3, S = 2

VGG 16


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 10
convolution + ReLU
max pooling
fully Connected + SIGMOID
softmax

Sigmoid

Activation Functions



Sigmoid

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

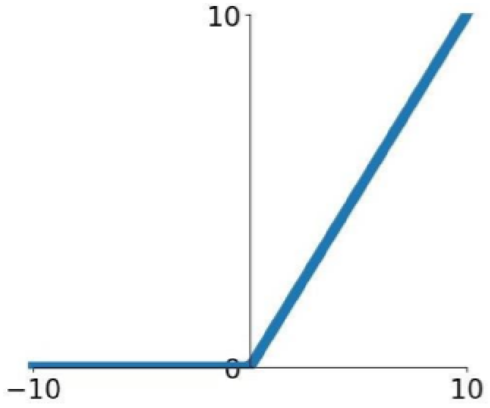
- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

3 problems:

1. Saturated neurons “kill” the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

ReLu Activation Function

Activation Functions



ReLU
(Rectified Linear Unit)

- Computes **f(x) = max(0,x)**

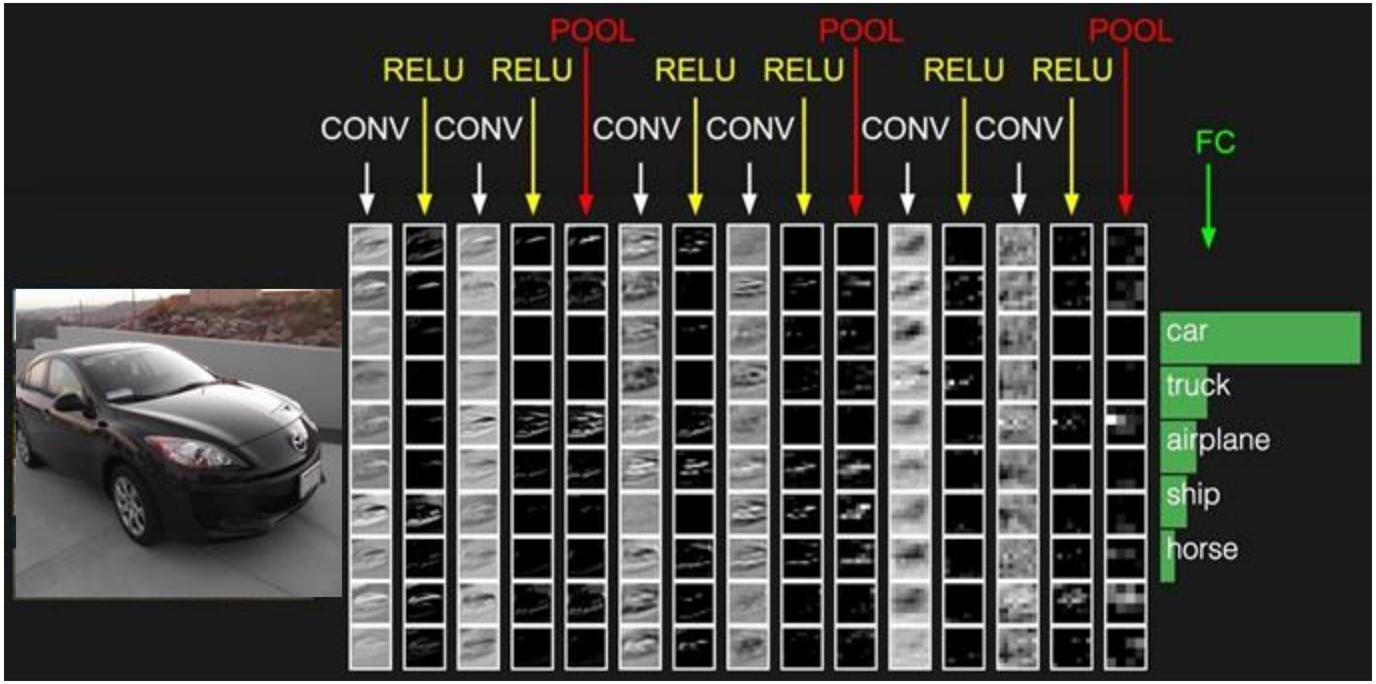
- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)
- Actually more biologically plausible than sigmoid

- Not zero-centered output
- An annoyance:

hint: what is the gradient when x < 0?

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Demo: http://cs231n.stanford.edu/

Developing your CNN for image classification

- ✓ Conv layer, Relu
- ✓ Conv layer, Relu
- ✓ Max pool
- ✓ Conv layer, Relu
- ✓ Conv layer, Relu
- ✓ Max pool
- ✓ Flatten
- ✓ FC1, Relu
- ✓ FC2, Relu
- ✓ FC3, Relu
- ✓ Softmax

TODO Task 1

- ✓ Senior RAs will guide you through the CNN codes for above architecture
- ✓ Run your code
- ✓ Produce the train and validation loss curves
- ✓ Test and produce your confusion matrix and evaluation metrics

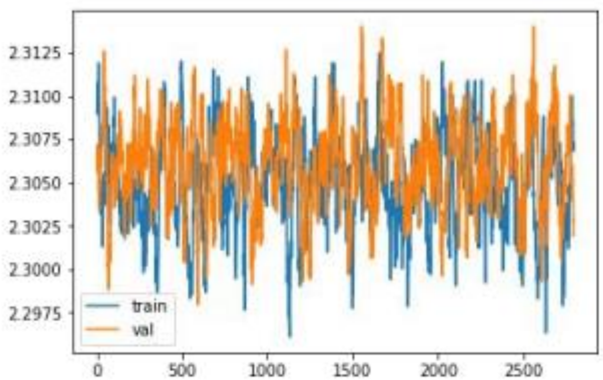
Babysitting your CNN

Training Process

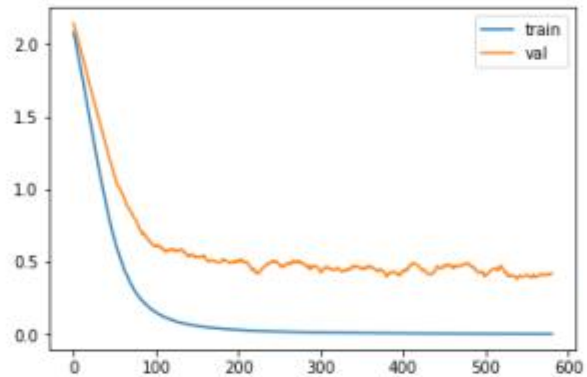
Mini-batch Stochastic Gradient Decent :

- ✓ Sample a batch of data
- ✓ Forward prop it through the graph (network)
- ✓ Calculate loss
- ✓ Backprop to calculate the gradients
- ✓ Update the parameters using the gradient

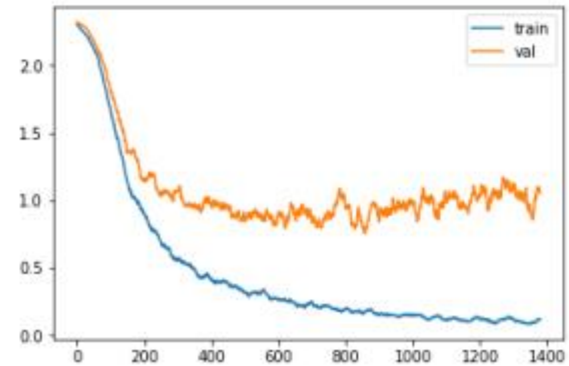
Loss curve investigation



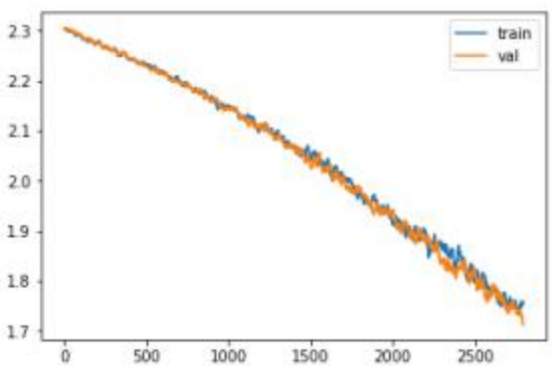
Not learning: gradients not applied to weights



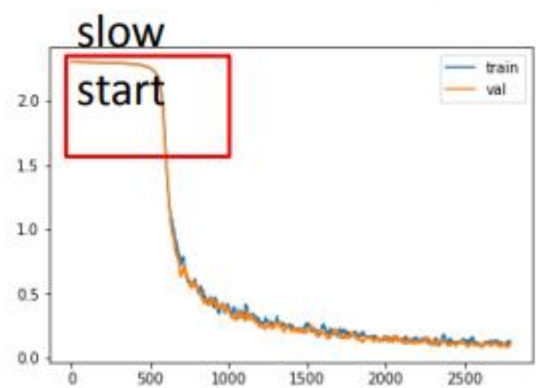
Overfit: model too large/dataset too small



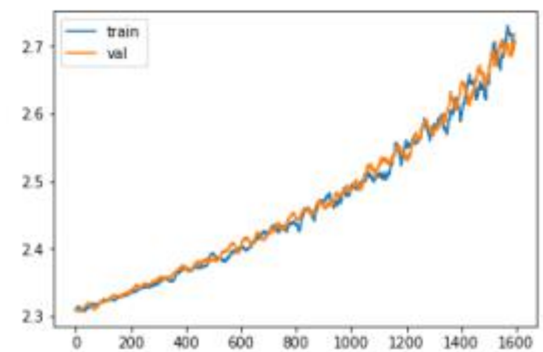
More extreme case of overfitting



Not converged yet: need longer training



Slow start: initialization weights too small

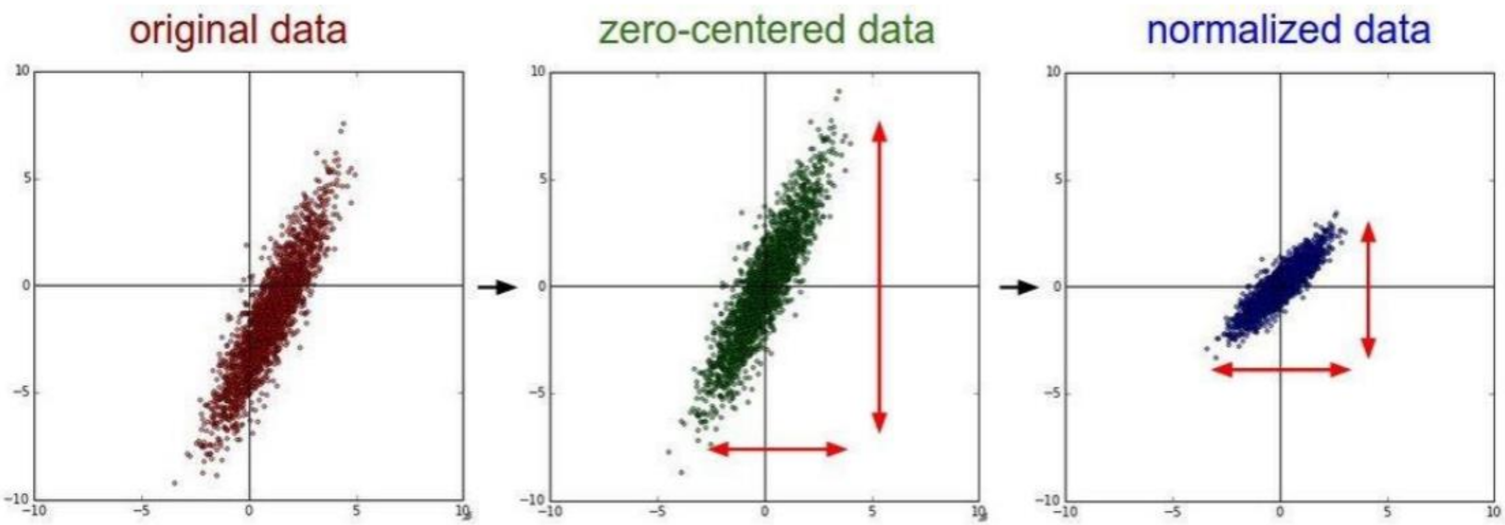


Applied the negative of gradients

Loss curve investigation

- Loss curve is one powerful indicator when debugging NNs
- Abnormal loss curves can be caused by
  - Wrong implementation of data loading
  - Wrong implementation/choice of losses
  - Optimizer problems
  - Suboptimal hyper-parameters

Normalize your Data



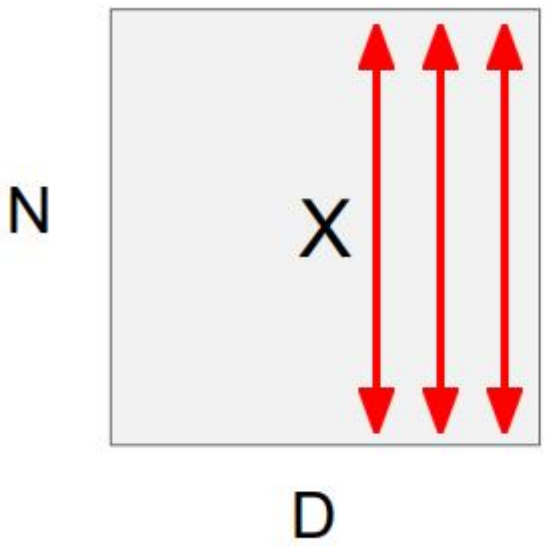

```
X -= np.mean(X, axis = 0)
```

```
X /= np.std(X, axis = 0)
```

(Assume X [NxD] is data matrix,
each example in a row)

Batch Normalization

"you want zero-mean unit-variance activations? just make them so."

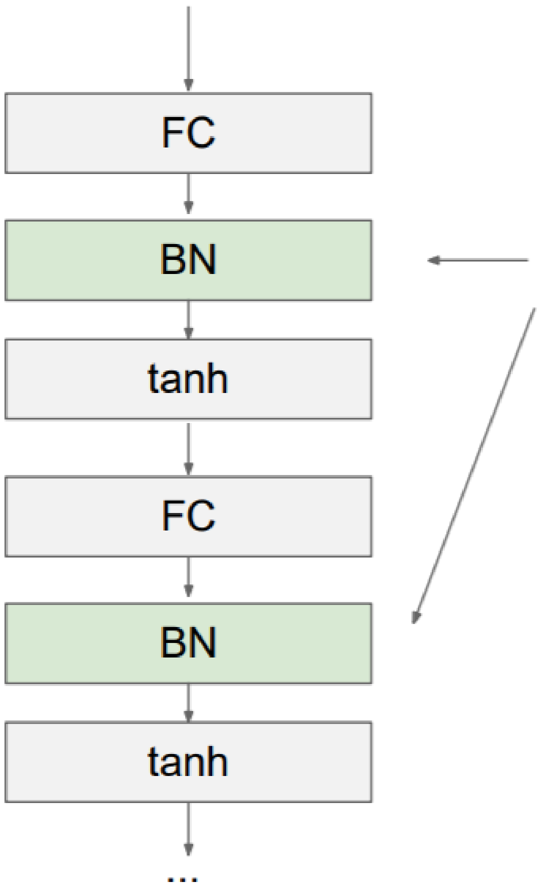


1. compute the empirical mean and variance independently for each dimension.

2. Normalize

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

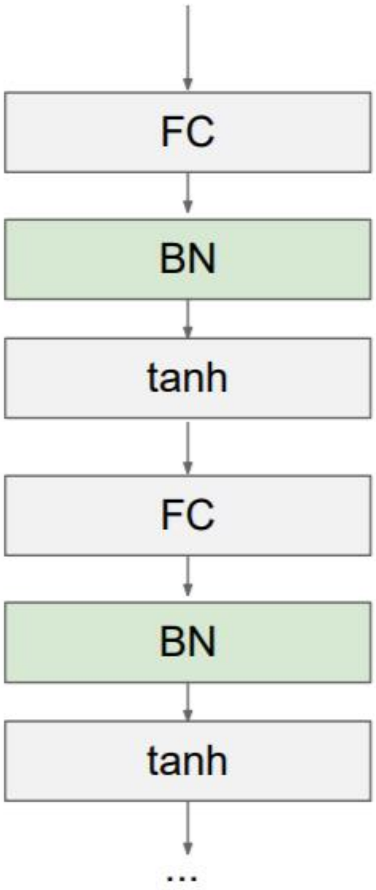
Batch Normalization



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

Problem: do we necessarily want a zero-mean unit-variance input?

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

Normalize:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

And then allow the network to squash the range if it wants to:

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}$$

Note, the network can learn:

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$

$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

to recover the identity mapping.

TODO Task 2

- ✓ Generally, fully connected layers are followed by sigmoid function. Add **sigmoid activation function** after each linear layer. Remove the Relu function for linear layers (architecture 2)
  - ✓ Run your code
  - ✓ Produce the train and validation loss curves
  - ✓ Test and produce your confusion matrix and evaluation metrics
- ✓ Add one convolution layer and Relu layer just before the flattening (architecture 3)
  - ✓ Run your code
  - ✓ Produce the train and validation loss curves
  - ✓ Test and produce your confusion matrix and evaluation metrics
- ✓ Add batch normalizations in the first two FC layers before sigmoid activation function (architecture 4)
  - ✓ For each of the fc layers (just before the sigmoid)
  - ✓ Run your code
  - ✓ Produce the train and validation loss curves
  - ✓ Test and produce your confusion matrix and evaluation metrics

TODO Task 3

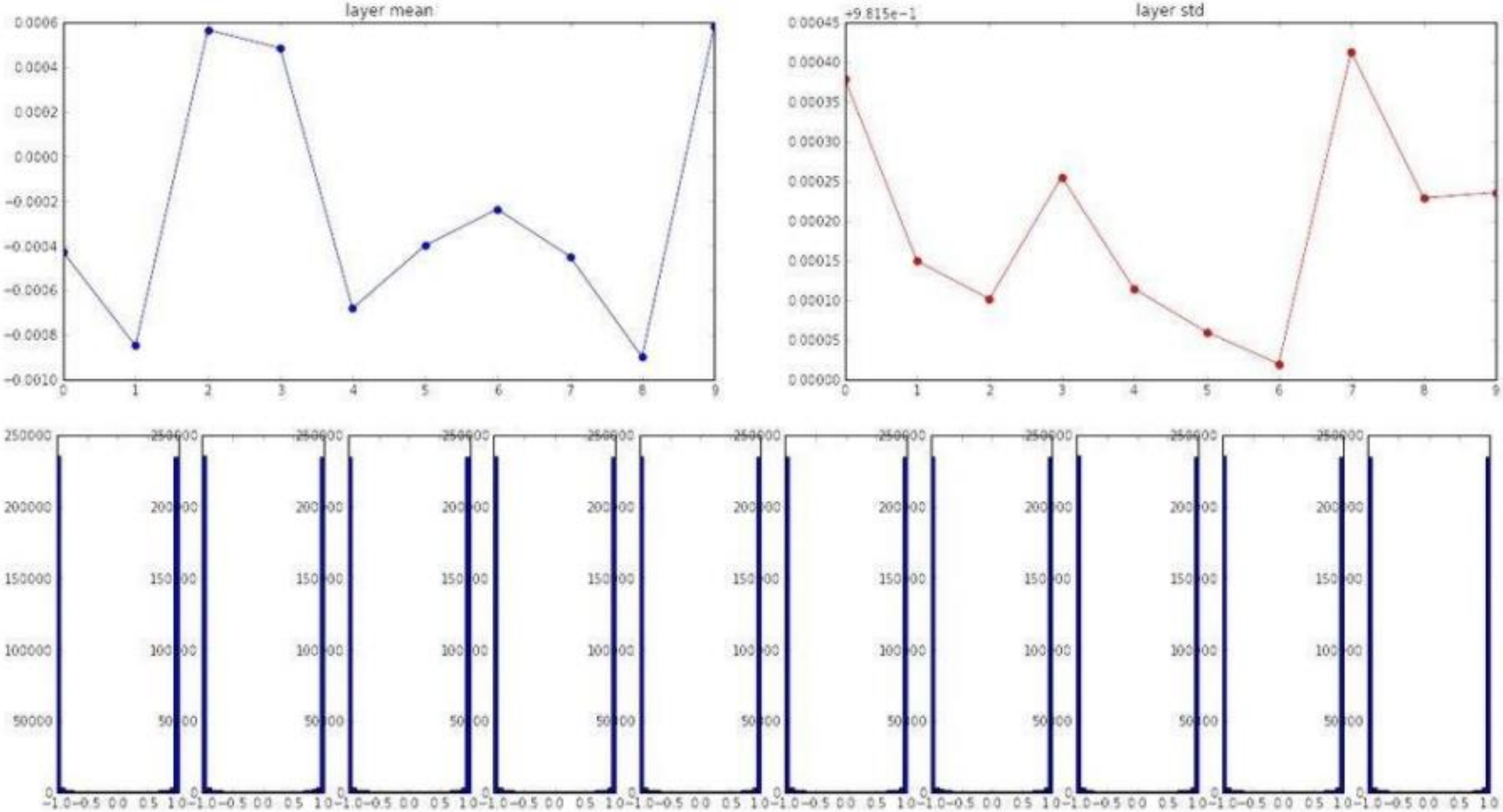
Create a table to compare your results for four architecture

- Original CNN (architecture 1)
- CNN with sigmoid (architecture 2)
- CNN with sigmoid and one extra conv + relu layer (architecture 3)
- Architecture 3 With BN layers (architecture 4)

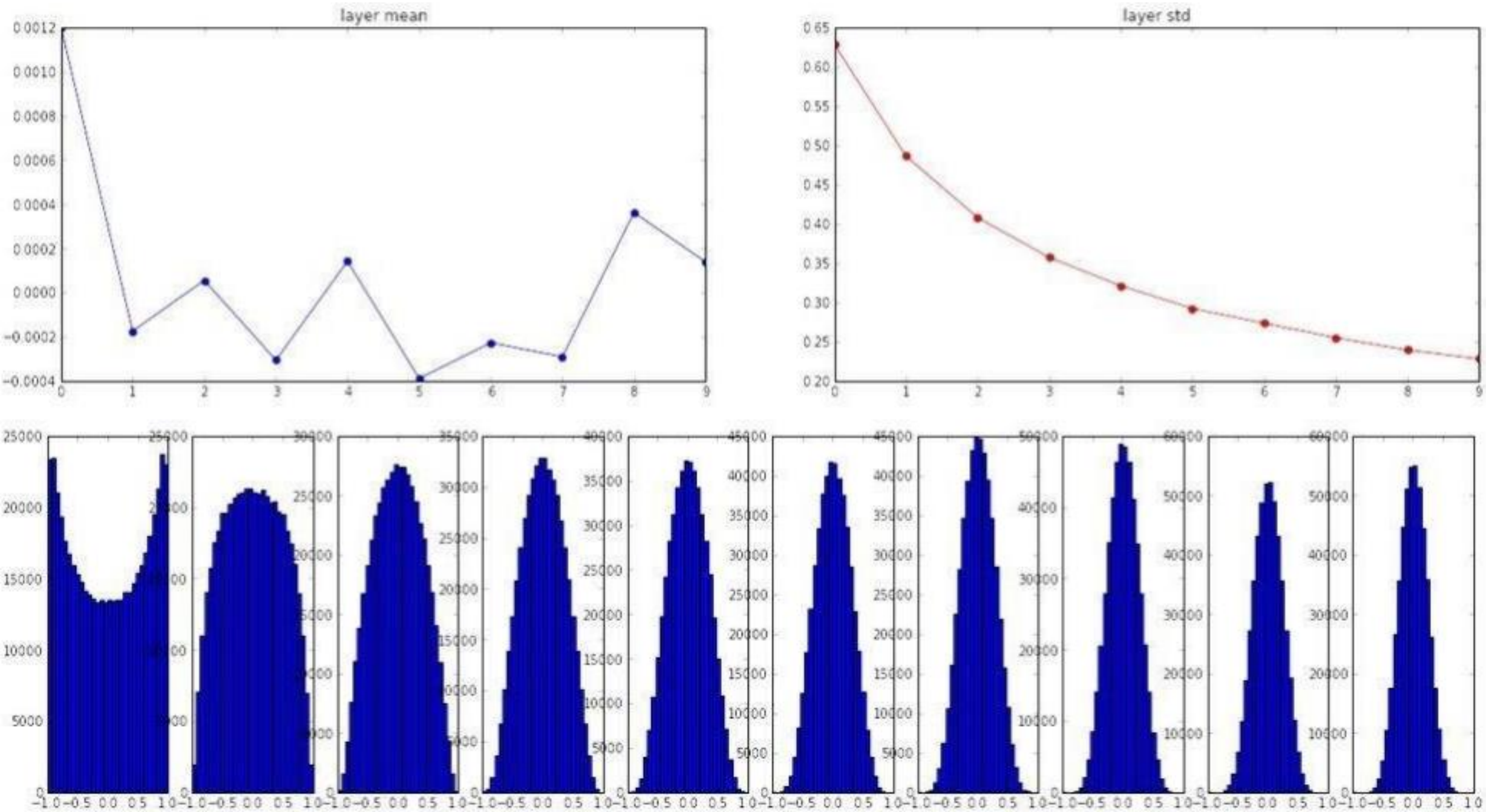
Weight Initialization

Random Initialization
Xavier Initialization [Glorot et al., 2010]
- Reasonable initialization

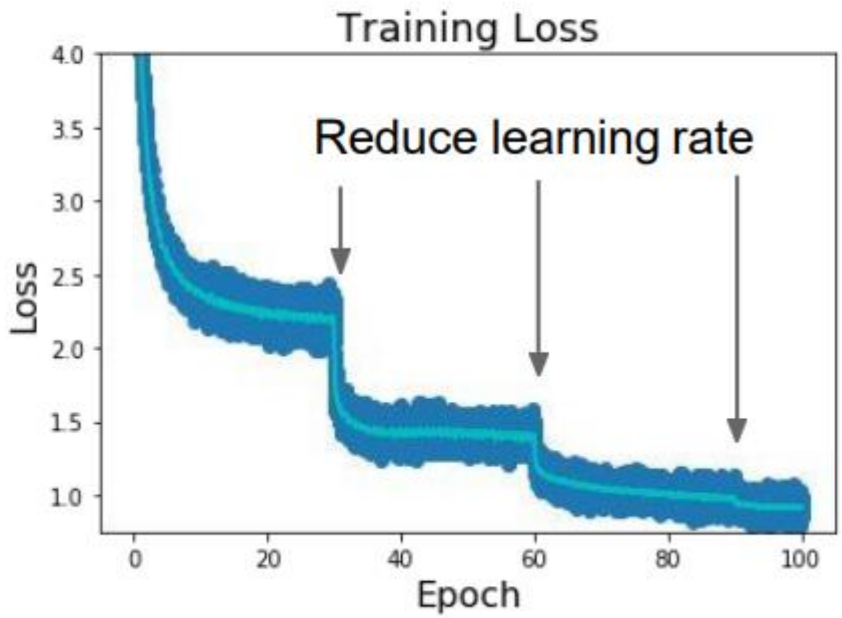
Random Initialization



Xavier Initialization



Learning rate decay



Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Learning rate decay

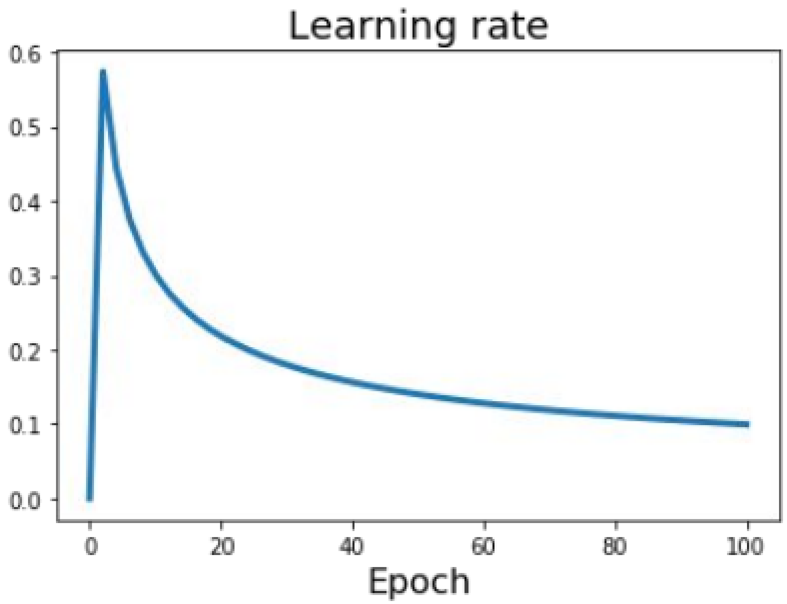
Cosine: $\alpha_t = \frac{1}{2}\alpha_0\left(1 + \cos(t\pi/T)\right)$

Linear: $\alpha_t = \alpha_0(1 - t/T)$

Inverse sqrt: $\alpha_t = \alpha_0/\sqrt{t}$

α_0 : Initial learning rate
α_t : Learning rate at epoch t
T : Total number of epochs

Learning Rate Decay: Linear Warmup



High initial learning rates can make loss explode; linearly increasing learning rate from 0 over the first ~5000 iterations can prevent this

Empirical rule of thumb: If you increase the batch size by N, also scale the initial learning rate by N

Goyal et al, “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”, arXiv 2017

Regularization: Add term to loss

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

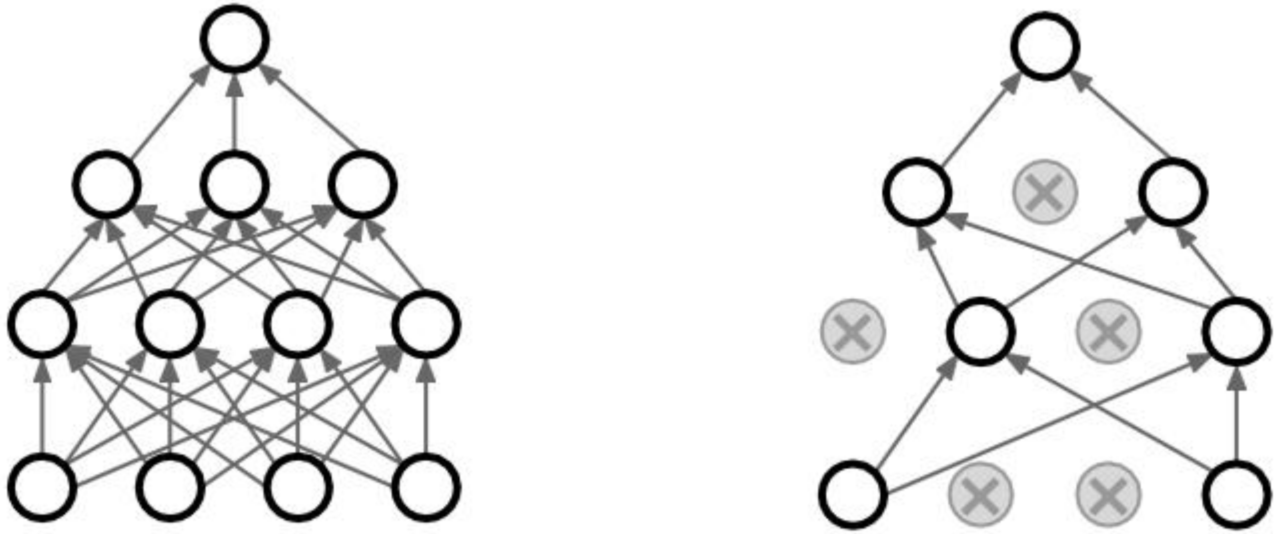
L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ (Weight decay)

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Regularization: Dropout

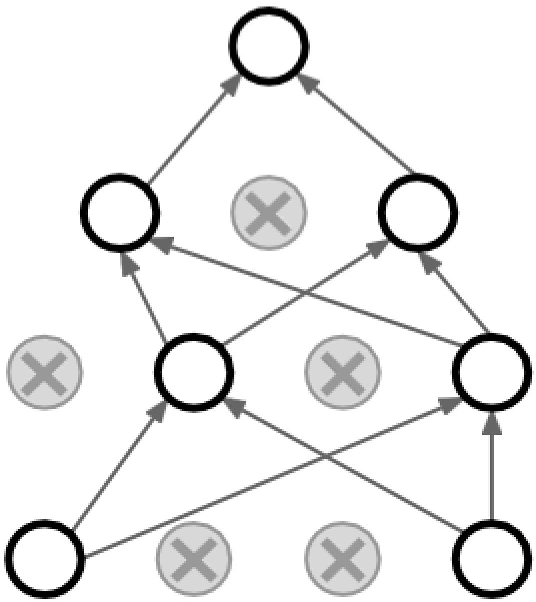
In each forward pass, randomly set some neurons to zero
Probability of dropping is a hyperparameter; 0.5 is common



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

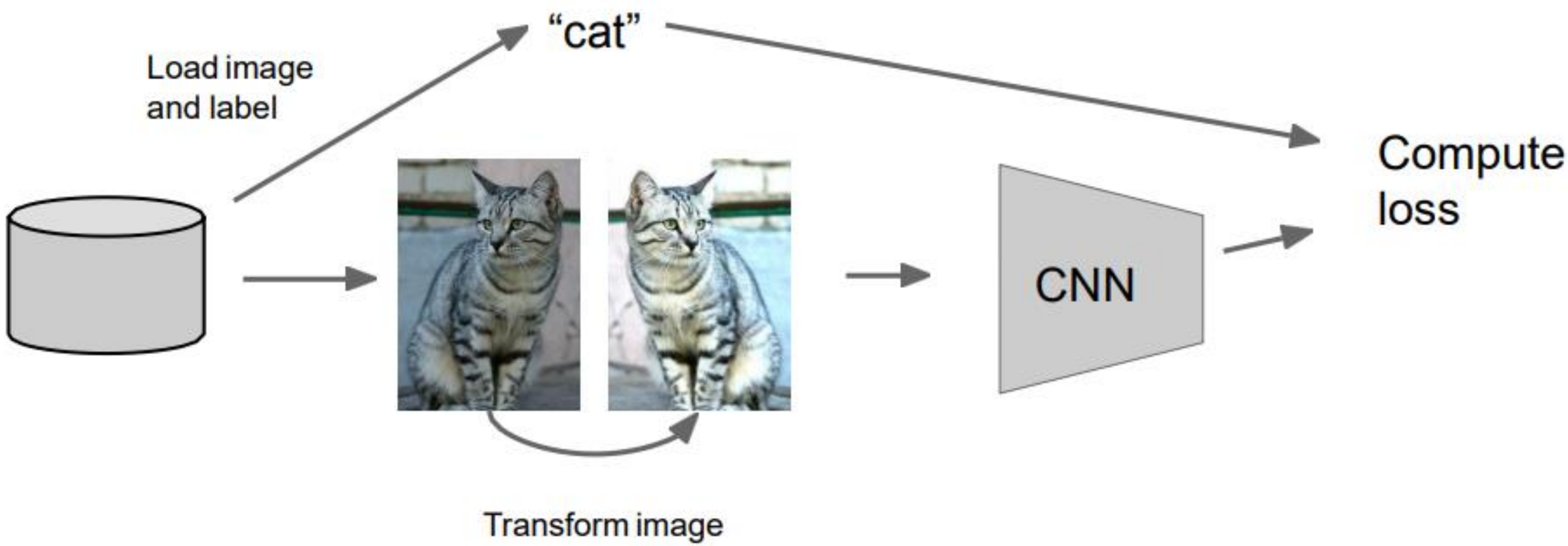
Regularization: Dropout

How can this possibly be a good idea?

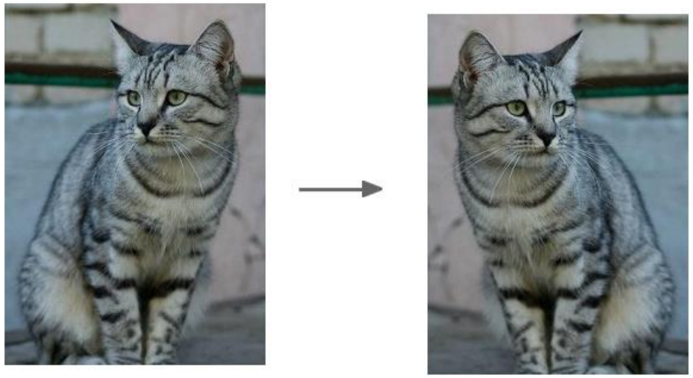


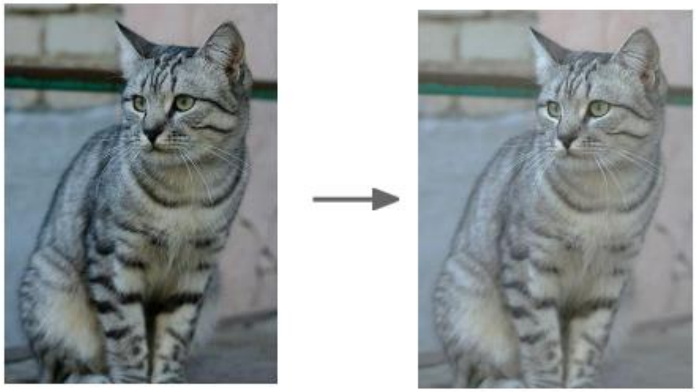
Forces the network to have a redundant representation;
Prevents co-adaptation of features

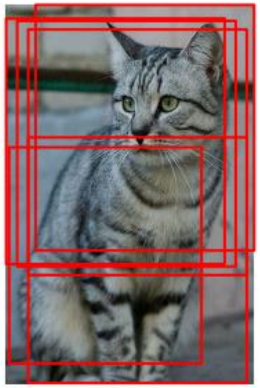
Data Augmentation



Data Augmentation







Choosing Hyper parameter

Step 1: Check initial loss

Step 2: Overfit a small sample

Step 3: Find LR that makes loss go down Use the architecture from the previous step, use all training data, turn on small weight decay, find a learning rate that makes the loss drop significantly within ~100 iterations

Good learning rates to try: 1e-1, 1e-2, 1e-3, 1e-4

Hyper-parameter Tuning

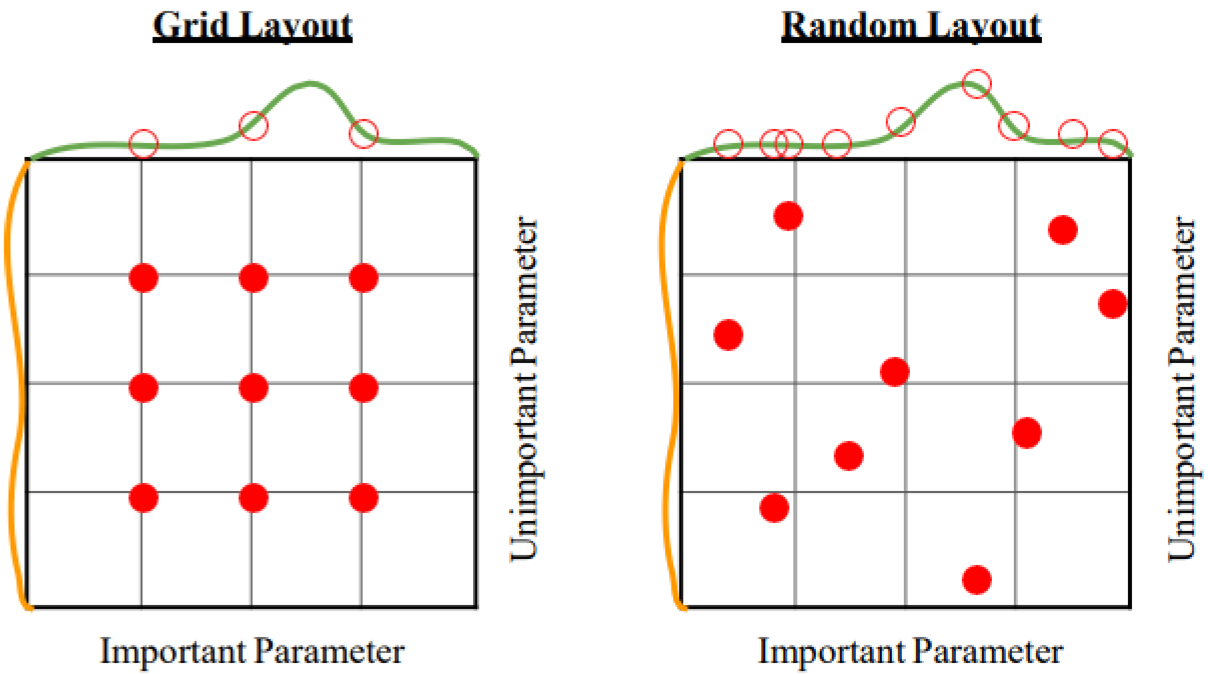


Illustration of Bergstra et al., 2012 by Shayne Longpre, copyright CS231n 2017

TODO Task 4

- ✓ Take the best performing architecture from the last task
- ✓ Add dropout in the first two FC layers after the activation functions (probability: 0.5
- ✓ Initialize your weight with Xavier
- ✓ Hyper parameter options:
  - ✓ Batch Size: 256, 512, 1024
  - ✓ Learning Rate: 0.0001, 0.001
  - ✓ Learning rate scheduler: Linear, Cosine, sqrt

- ✓ Run training
- ✓ Produce the train and validation loss curves
- ✓ Test and produce your confusion matrix and evaluation metrics

CIFAR10 Image classification using pretrained VGG16

- ✓ VGG 16 weights are trained in Image nets
- ✓ We will finetune the weights for CIFAR10
- ✓ Performance Evaluation

Text Processing (NLP)

Downstream tasks:

- Emotion recognition from sentences
- Sentiment analysis from reviews
- Automatic Essay grading
- Question Answering AI bot (Chat GPT, Gemini, Claude)

How to represent a sentence

University of Virginia

From Wikipedia, the free encyclopedia

The **University of Virginia** (**UVA** or **U.Va.**), often referred to as simply **Virginia**, is a public research university in Charlottesville, Virginia. UVA is known for its historic foundations, student-run honor code, and secret societies.

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe. President Monroe was the sitting President of the United States at the time of the founding; Jefferson and Madison were the first two rectors. UVA was established in 1819, with its Academical Village and original courses of study conceived and designed entirely by Jefferson. UNESCO designated it a World Heritage Site in 1987, an honor shared with nearby Monticello.[4]

The first university of the American South elected to the Association of American Universities in 1904, UVA is classified as *Very High Research Activity* in the Carnegie Classification. The university is affiliated with 7 Nobel Laureates, and has produced 7 NASA astronauts, 7 Marshall Scholars, 4 Churchill Scholars, 29 Truman Scholars, and 50 Rhodes Scholars, the most of any state-affiliated institution in the U.S.[5][6][7] Supported in part by the Commonwealth, it receives far more funding from private sources than public, and its students come from all 50 states and 147 countries.[2][8][9] It also operates a small liberal arts branch campus in the far southwestern corner of the state.

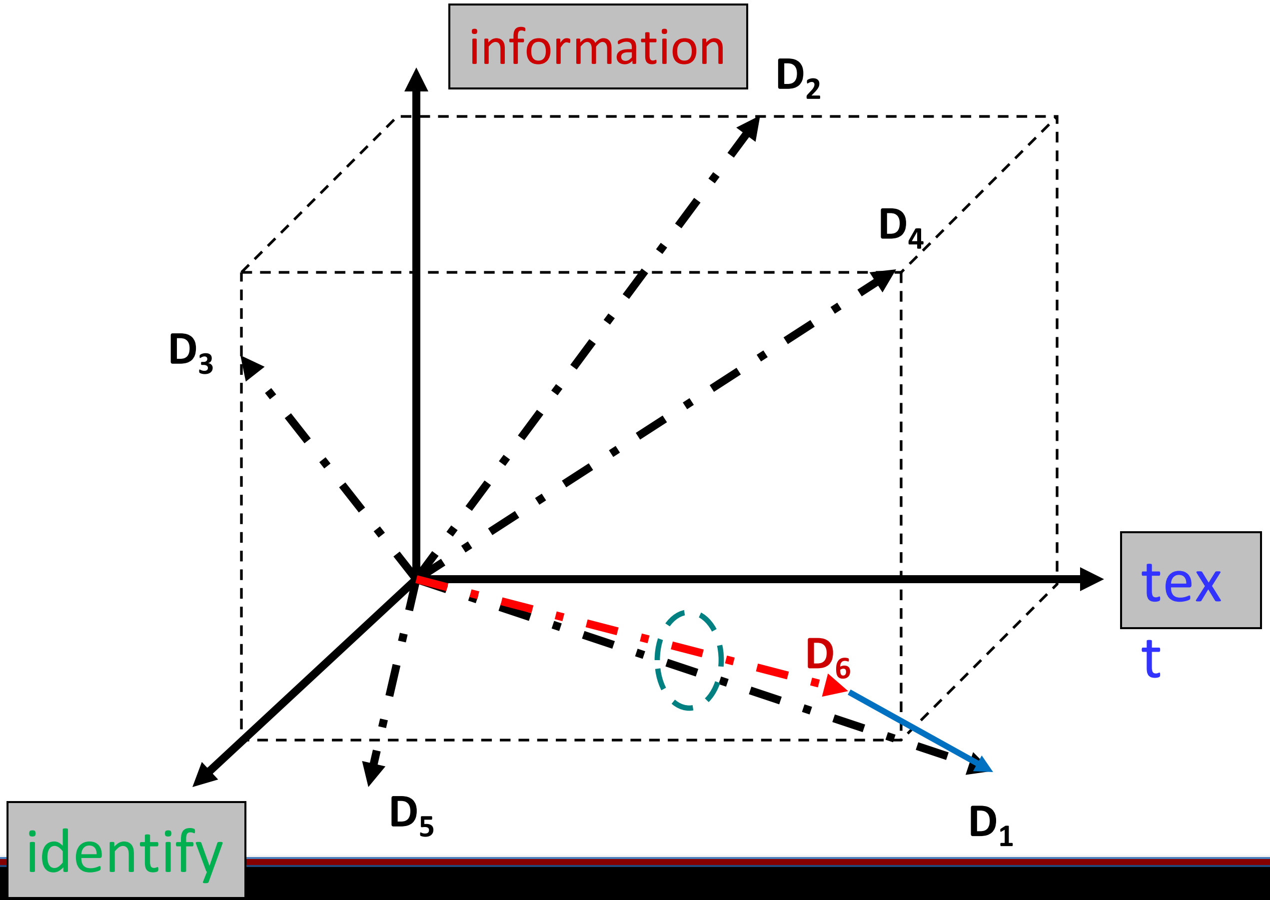
Bag-of-Words representation

- Term as the basis for vector space
  - Doc1: Text mining is to identify useful information.
  - Doc2: Useful information is mined from text.
  - Doc3: Apple is delicious.

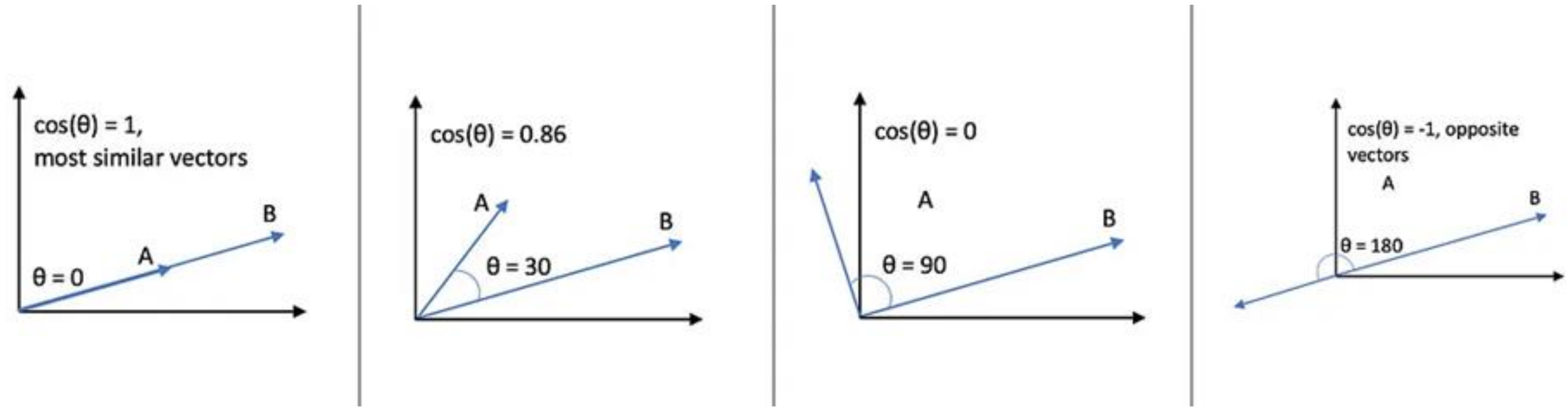
	text	information	identify	minin g	mined	is	usefu l	to	from	apple	deliciou s
Doc1	1	1	1	1	0	1	1	1	0	0	0
Doc2	1	1	0	0	1	1	1	0	1	0	0
Doc3	0	0	0	0	0	1	0	0	0	1	1

How to define a good similarity metric?

- Euclidean distance?



Similarity measurement



$$cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

Tokenization

- Break a stream of text into meaningful units
  - Tokens: words, phrases, symbols
    - **Input:** It’s not straight-forward to perform so-called “tokenization.”
    - **Output(1):** 'It’s', 'not', 'straight-forward', 'to', 'perform', 'so-called', '“tokenization.”'
    - **Output(2):** 'It', '’', 's', 'not', 'straight', '-', 'forward, 'to', 'perform', 'so', '-', 'called', ‘“', 'tokenization', '.', '”‘
  - Definition depends on language, corpus, or even context

Stemming

- Reduce inflected or derived words to their root form
  - Plurals, adverbs, inflected word forms
    - E.g., ladies -> lady, referring -> refer, forgotten -> forget
  - Bridge the vocabulary gap
  - Solutions (for English)
    - Porter stemmer: patterns of vowel-consonant sequence
    - Krovetz stemmer: morphological rules
  - Risk: lose precise meaning of the word
    - E.g., lay -> lie (a false statement? or be in a horizontal position?)

Stopwords

- Useless words for document analysis
  - Not all words are informative
  - Remove such words to reduce vocabulary size
  - No universal definition
  - Risk: break the original meaning and structure of text
    - E.g., this is not a good option -> option

      to be or not to be -> null

Stopwords

Nouns	Verbs	Adjectives	Prepositions	Others
1. time	1. be	1. good	1. to	1. the
2. person	2. have	2. new	2. of	2. and
3. year	3. do	3. first	3. in	3. a
4. way	4. say	4. last	4. for	4. that
5. day	5. get	5. long	5. on	5. I
6. thing	6. make	6. great	6. with	6. it
7. man	7. go	7. little	7. at	7. not
8. world	8. know	8. own	8. by	8. he
9. life	9. take	9. other	9. from	9. as
10. hand	10. see	10. old	10. up	10. you
11. part	11. come	11. right	11. about	11. this
12. child	12. think	12. big	12. into	12. but
13. eye	13. look	13. high	13. over	13. his
14. woman	14. want	14. different	14. after	14. they
15. place	15. give	15. small	15. beneath	15. her
16. work	16. use	16. large	16. under	16. she
17. week	17. find	17. next	17. above	17. or
18. case	18. tell	18. early		18. an
19. point	19. ask	19. young		19. will
20. government	20. work	20. important		20. my
21. company	21. seem	21. few		21. one
22. number	22. feel	22. public		22. all
23. group	23. try	23. bad		23. would
24. problem	24. leave	24. same		24. there
25. fact	25. call	25. able		25. their

Bag-of-Words representation

	text	information	identify	minin g	mined	is	usefu l	to	from	apple	deliciou s
Doc1	1	1	1	1	0	1	1	1	0	0	0
Doc2	1	1	0	0	1	1	1	0	1	0	0
Doc3	0	0	0	0	0	1	0	0	0	1	1

- Assumption
  - Words are independent from each other
- Pros
  - Simple
- Cons
  - Basis vectors are clearly not linearly independent!
  - Grammar and order are missing

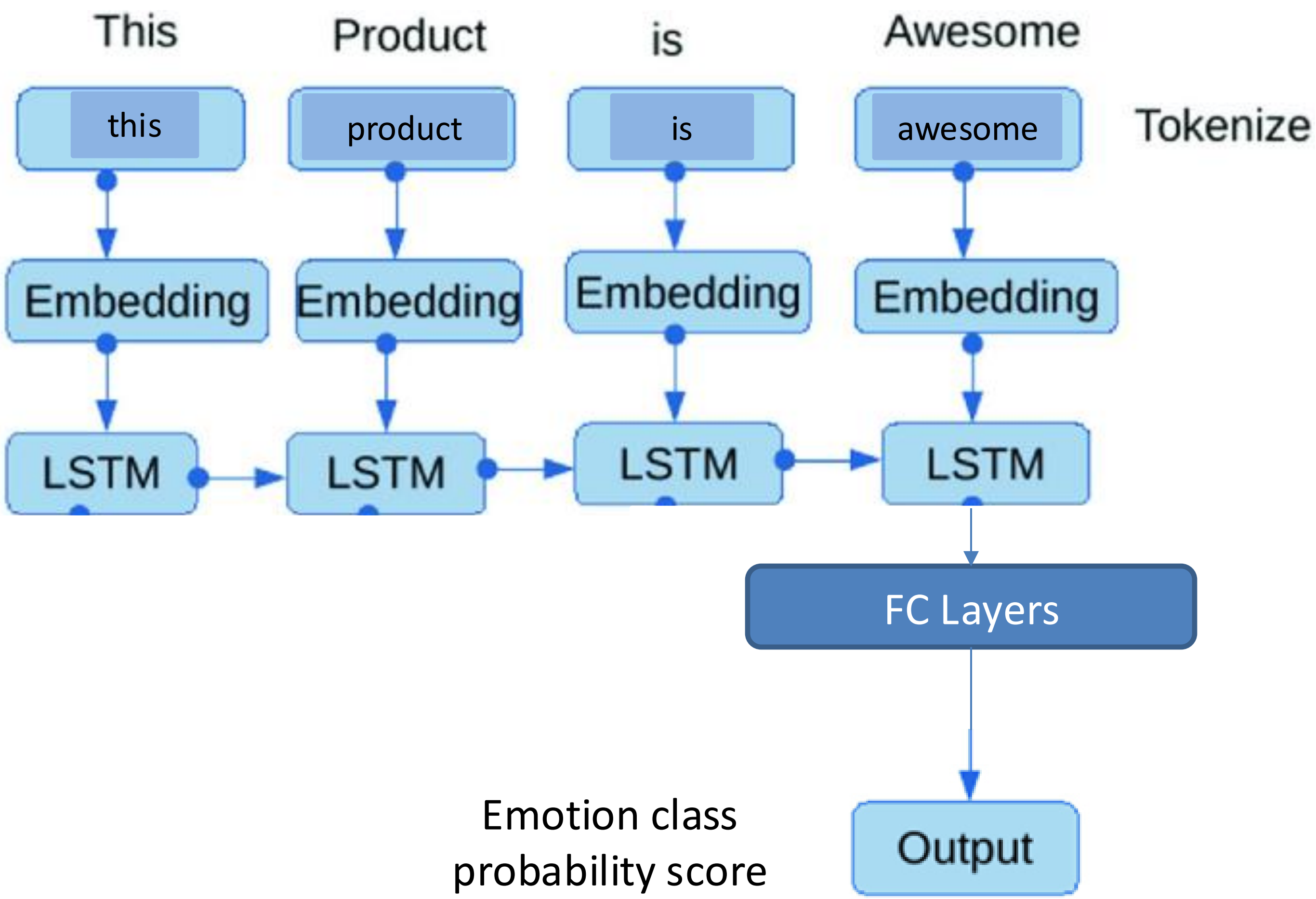
Word Embedding

- One Hot
  - Word Dimension size will increase with vocabulary size
  - Slow
  - Words are not orthogonal each other
  - Word synonyms are orthogonal (not preferred)
- Fixed dimension embedding
  - Built-in torch embedding
  - Word2vec

Sequential Modeling with LSTM

- I am playing cricket.
- Am I playing cricket?

Sequential Modeling with LSTM



Text Processing with one-hot encoding

Text Processing with built-in torch embedding

TODO Task 5

- Use **word2vec** for word embedding for your LSTM based emotion recognition model
- **word2vec** library is already downloaded into your runtime memory
- Show confusion matrix, average F1 measure, Compare your performance with torch embedding

Contextual Embedding

Contextual Embeddings

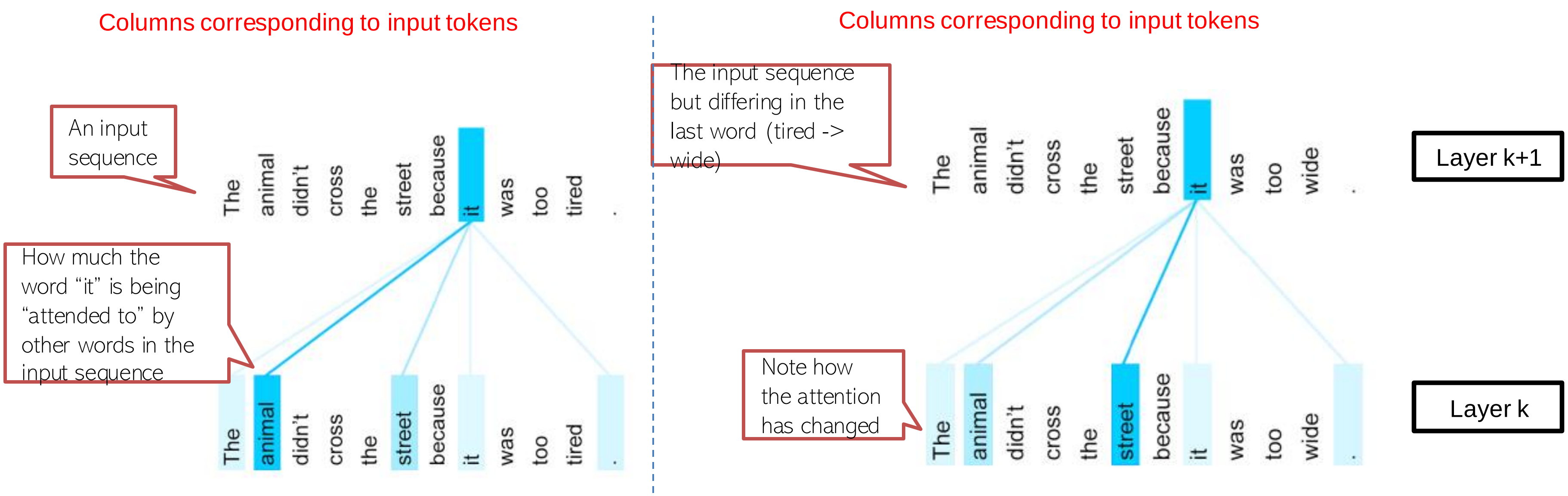
`The animal didn't cross the road because it`

What should be the properties of "it"?

`The animal didn't cross the road because it was too` **`tired`**
`The animal didn't cross the road because it was too` **`wide`**

At this point in the sentence, it's probably referring to either the `chicken` or the street

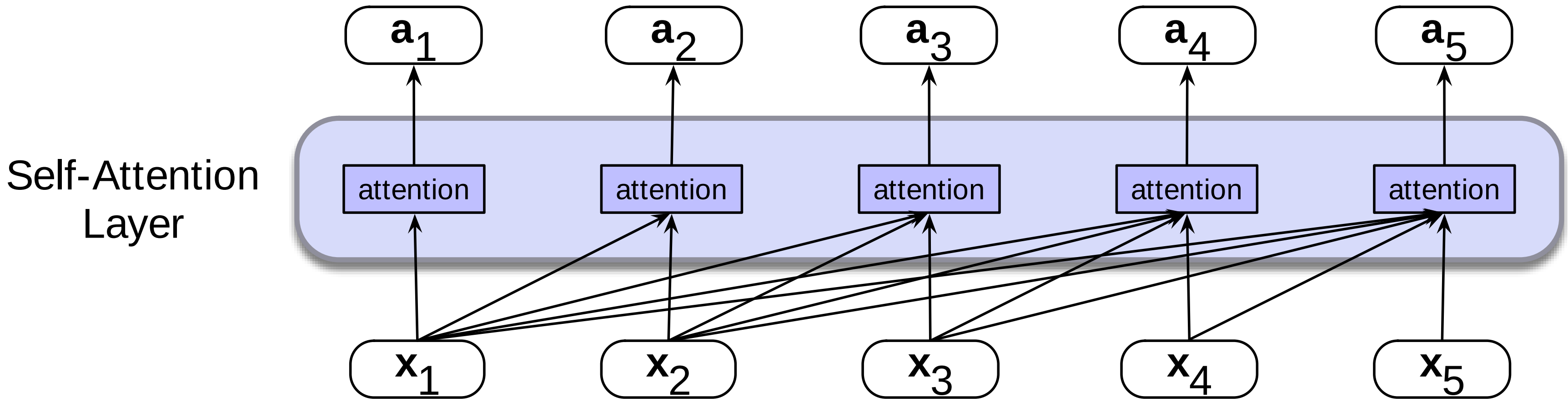
Intuition of attention: (Self-Attention)



- Attention helps capture the context better and in a much more "global" manner
  - "Global": Long ranges captures and in both directions (previous and ahead)

Example credit: https://blog.research.google/2017/08/transformer-novel-neural-network.html

Attention is left-to-right (a simpler view)



Simplified version of attention: a sum of prior words weighted by their similarity with the current word

Given a sequence of token embeddings:

$$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \mathbf{x}_4 \quad \mathbf{x}_5 \quad \mathbf{x}_6 \quad \mathbf{x}_7 \quad \mathbf{x}_i$$

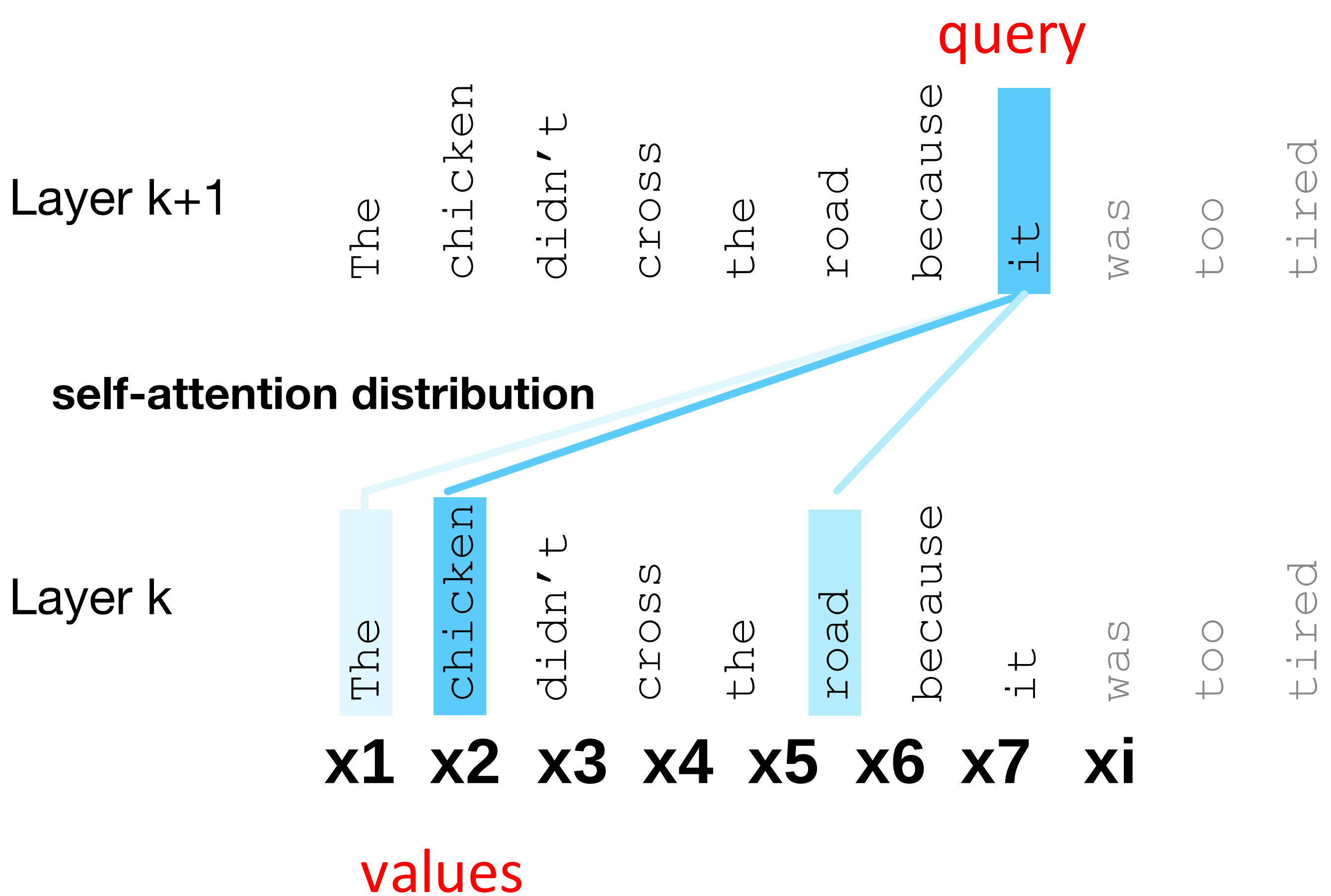
Produce: $\mathbf{a}_i$ = a weighted sum of $\mathbf{x}_1$ through $\mathbf{x}_7$ (and $\mathbf{x}_i$)
Weighted by their similarity to $\mathbf{x}_i$

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$

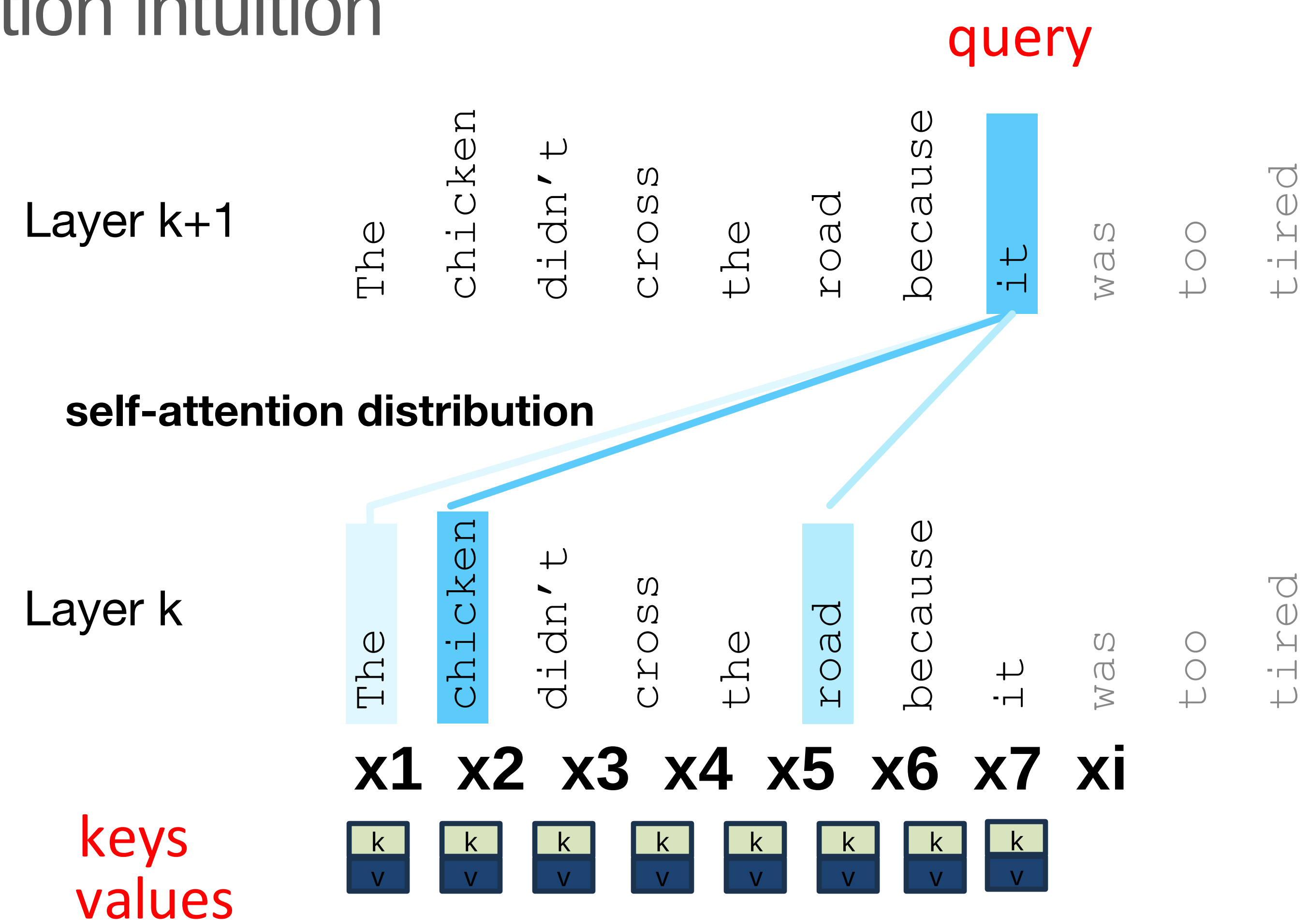
$$a_{ij} = \text{softmax}(\text{score}(x_i, x_j)) \quad j \le i$$

$$\mathbf{a}_i = \sum_{j \le i} \alpha_{ij} \mathbf{x}_j$$

Attention intuition



Attention intuition



Final equations for one attention head

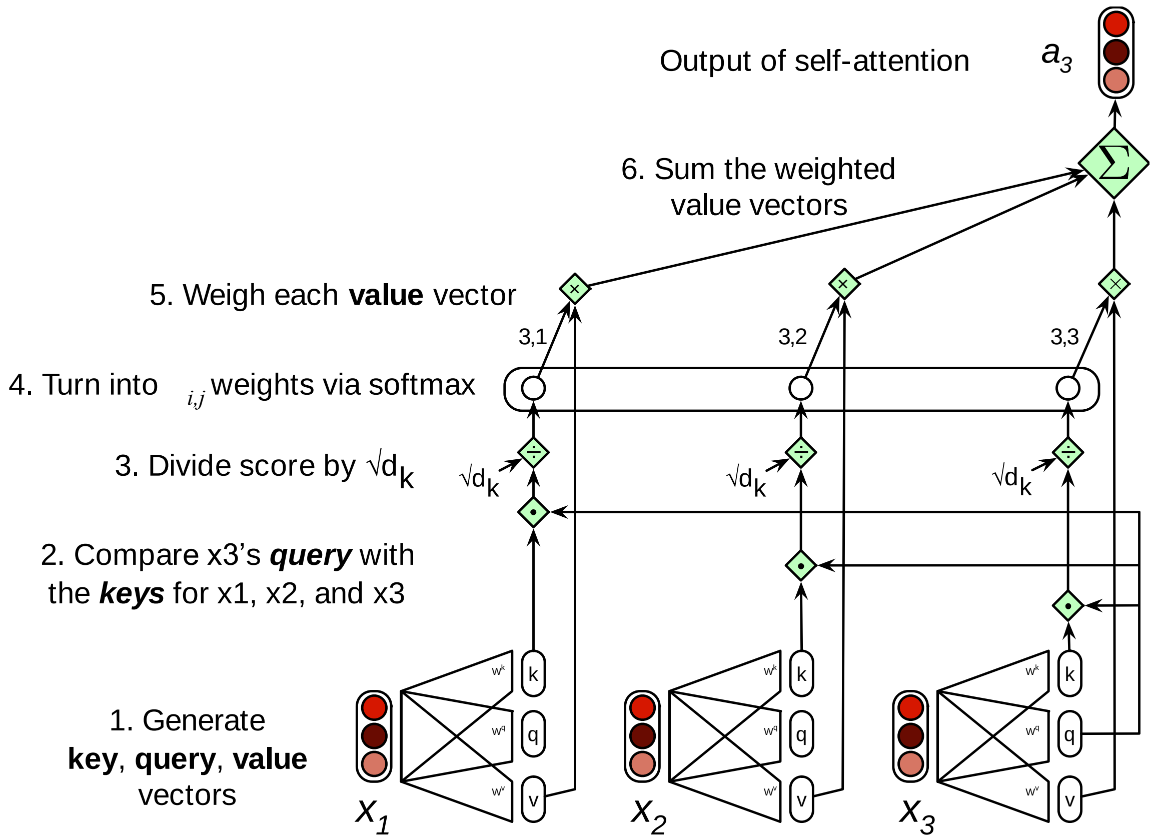
$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{V}}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

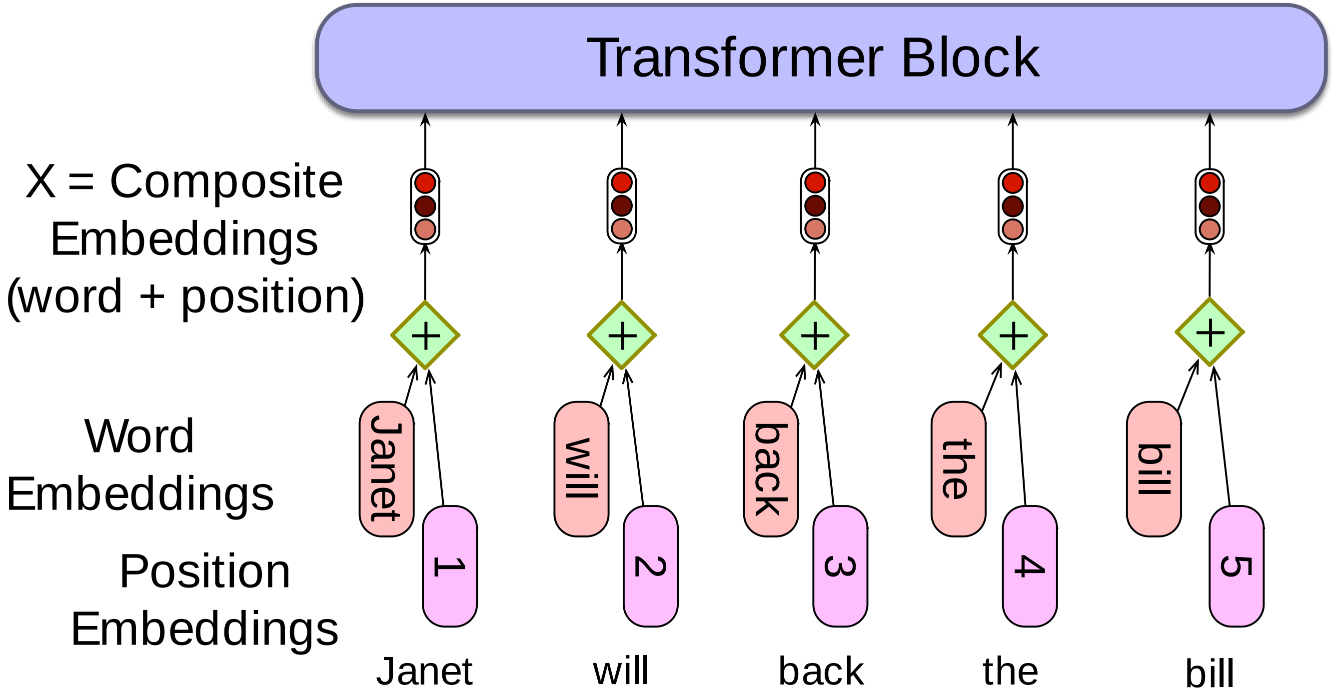
$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i$$

$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$

Calculating the value of a3



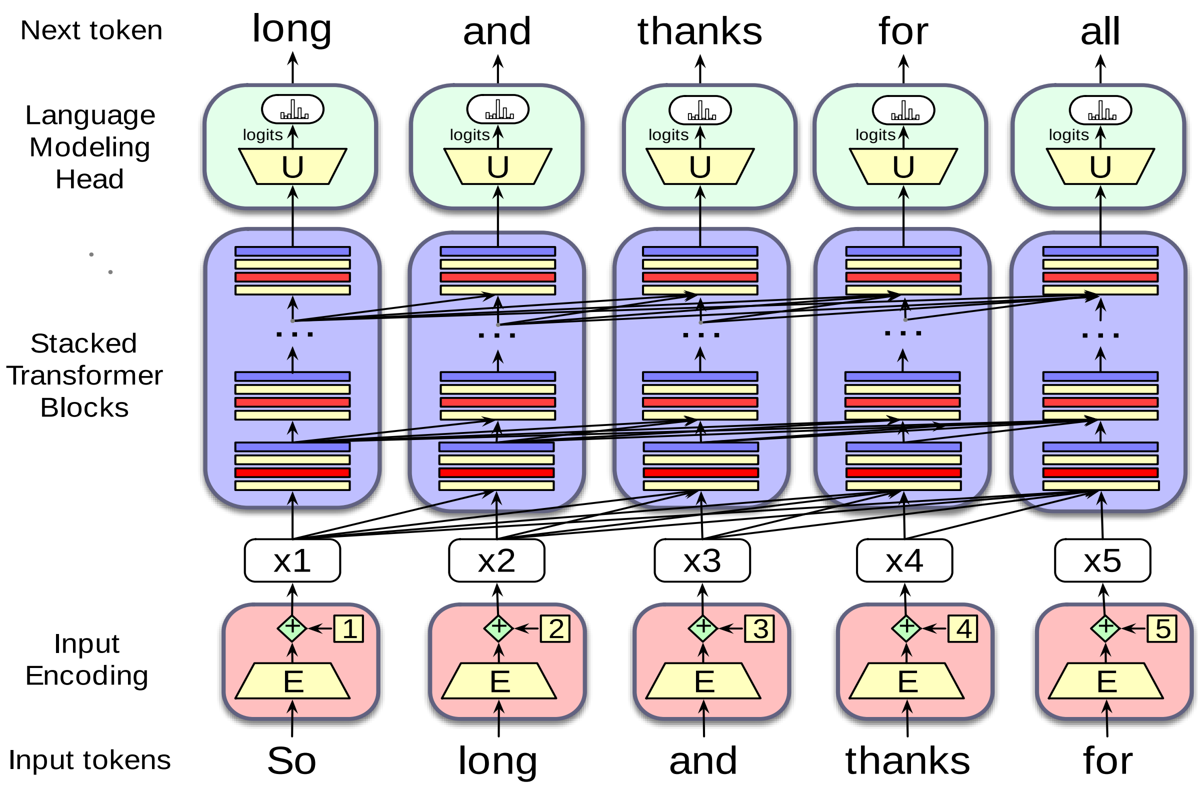
Position Embedding

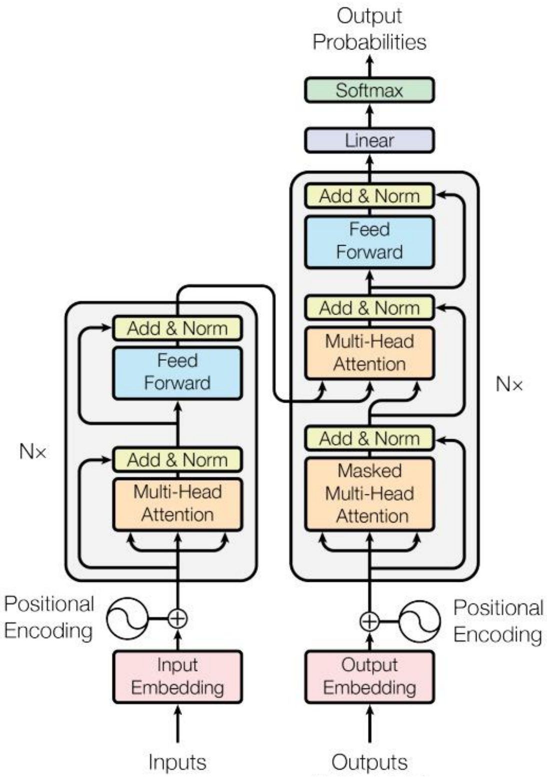


If embedding has a dimensionality of 4:

Transformer Architecture

Let's consider the embeddings for an individual word from a particular layer for next word prediction

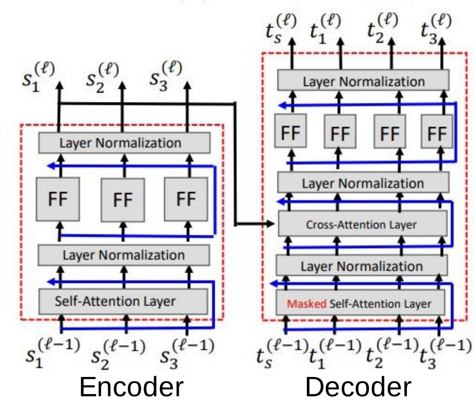




One transformer Block

Popular Transformer Variants: BERT and GPT

- The standard transformer architecture is an encoder-decoder model
- Some models use just the encoder <u>or</u> the decoder of the transformer
- BERT (Bidirectional Encoder Representations from Transformers)
  - Basic BERT can be learned to encoder token sequences
- GPT (Generative Pretrained Transformer)
  - Basic GPT can be used to <u>generate</u> token sequences similar to its training data

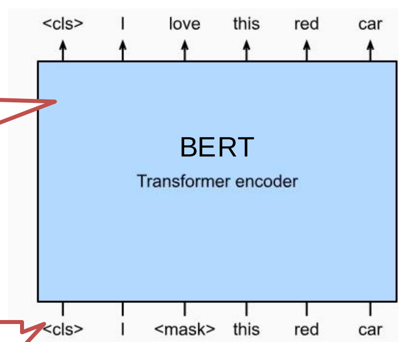


This encoder can be used for other tasks by fine-tuning

A transformer which contains only the encoder

Trained unsupervisedly using a missing token prediction objective

This is just start of sentence token

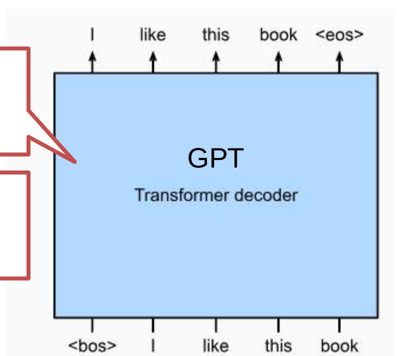


Missing token which BERT tries to predict

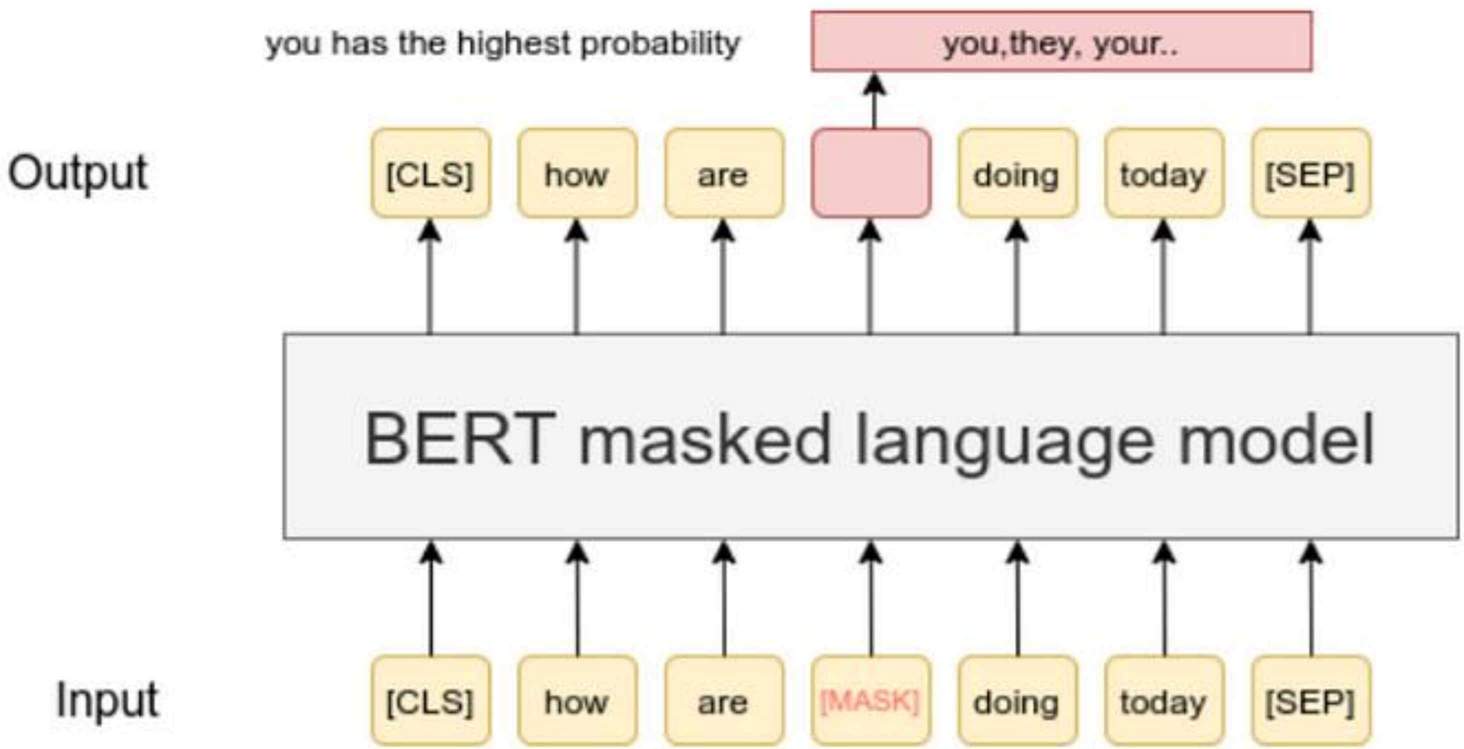
Also, no cross-attention since there is no encoder

A transformer which contains only the decoder

Pre-trained using a next token prediction objective

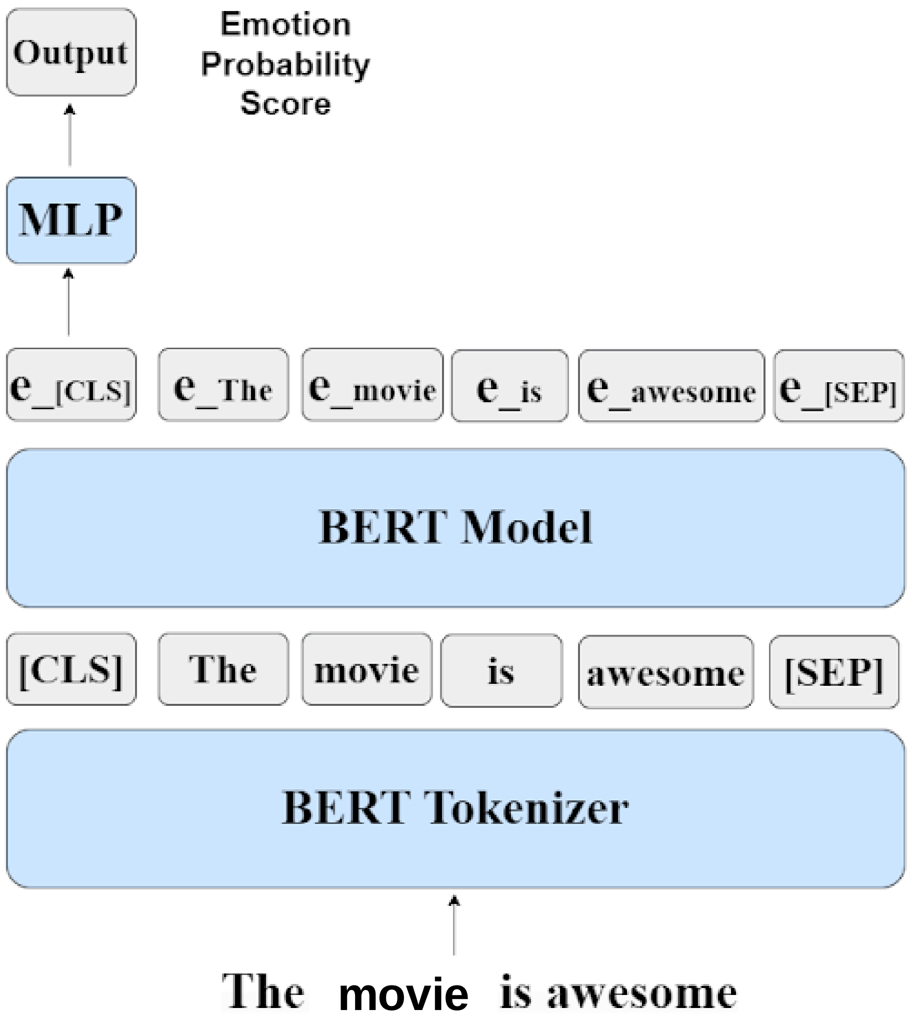


BERT Pretraining



Blog: https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51

Emotion recognition using pretrained BERT



TODO Task 6: Identify the Most Attentive Tokens for Each Class in the Test Dataset

- The BERT model computes attention scores for each word in every layer, which influence its predictions.
- After training or fine-tuning the BERT model on the Emotion dataset, the model is expected to focus more on content-specific words relevant to the target class.
- **Your task is to extract the most attentive tokens based on the BERT attention scores.**
- For instance, **in the "Happy" class, the most attentive tokens could be "joyful," "wow," or "pleased."**

Detail task description:

https://iubedubd-my.sharepoint.com/:b:/g/personal/akmmrahman_iub_edu_bd/EciUyZiLg-pHtQ64a32Ox6cB9KlhKOrMyQyCUc18hfay1A?e=BE4UVd