L1 – introduction + history of Computer Vision

L2 – image classification talking about popular datasets like :

-MNIST

-CIFAR10

-ImageNet

-MIT Places

-Omniglot

-k-NN Algorithm

-train: prepare data set (Download + Preprocess)

-predict: let’s talk about image classification: u take each Test image and compute the Distance(Euclidian/Manhattan) between test and each image from the training set, sort it descendently and pick the top k distances and choose the predicted label via the most frequency class label.

-also use cross validation folding to choose the “best k”

- in case of a tie choose a strategy (e.g. pick the smallest label)

-curse of dimensionality

-distances between points in high dimensions are increasing exponentially

-computation inneficient(calculating distances between all points)

L3 – linear classifier(SVM)

-bias trick(not used in Neural Networks) – getting rid of the bias vector by concatenating it to the Weight matrices and adding a “1” to the X features Vector

-SVM (finding a hyperplan that best divide each class in sections)

-talking about CIFAR10 data set (50000 images) of (32,32,3) images of 10 classes(cat,car,frog,…)

- z = W \* X

W shape: (10,3072)

10 – 10 classes

3072 – image flattened(32 \* 32 \* 3 pixels)

X shape : (3072,1)

z Shape: (10,1) – 10 predicted scores for each class

-Visual viewpoint of the classifier (one template per class): the classifier tries to understand the template (shapes and colors of the background).

-ex: If it learns that most photos of horses are on a green backround and the horse is usually brown and tries to predict a car(not necessarily brown on a greenish backround it will most likely fail)

-Geometric point of the classifier: each row of the W matrix defines a hyperplan and the prediction consists in which hyperplan does the unseen image belong to.

-Loss functions: SVM loss – Multiclass SVM loss function – one vs all Approach

A black background with white text

AI-generated content may be incorrect.

Sj – score of each incorrect class

Syi – score of the correct class

Δ – 1

Total loss:

A black background with white symbols

AI-generated content may be incorrect.

Λ – lambda (L2 regularization term) prevents overfitting of data

Cross-Entropy Loss – interpret classifier scores as probabilities using SoftMax function

A black background with white text

AI-generated content may be incorrect.

L4

Batch Gradient Descent – determine the gradient by computing the derivative of the whole training set

-can be inneficient when having lots of data with lots of parameters to learn

-hyperparameters:

-learning rate

-number of steps(epochs)

-Weight initialiatization(currently important?!)

A close-up of a computer code

AI-generated content may be incorrect.

Stochastic Gradient Descent(SGD) – determining gradient by computing the derivative of minibatches of training set

-hyperparameters:

-learning rate

-number of steps(epochs)

-Weight initialiatization(currently important?!)

-Batch size(32/64/128)

-Data Sampling(not that important of image classification)

A close-up of a computer code

AI-generated content may be incorrect.

-choosing the learning rate is important because:

- if choosing a larger learning rate the model can overshoot over the globabl minimum and might never reach it

- if choosing a smaller learning rate the model can be slow and the number of epochs to converge could be so big that the computation power will not handle it

SGD + Momentum:

A screenshot of a computer

AI-generated content may be incorrect.

Adam – combines Momentum + Adaptive Learning Rate for faster convergence

L5

Feature transforms: applying complex functions for better understanding of different caracteristic of data such as:

- color representation

-edge direction & strength(Histogram of Oriented Gradients)

If we did not have any activation functions in our Neural Network the NN would simply be a linear model.

Applying activation functions breaks the liniarity of the model by using non-linear functions such as:

-ReLu

-Sigmoid

-Leaky ReLu

L6

Backpropagation: explained in CS229.txt from Stanford

Assignment 2:

D C

W shape (3073,10)

X shape (N,D)

(C,D) \* (N,D) = (C,D)

L7

Image shape : C x H x W

C -number of channels ex: RGB – C = 3

H=W=32 pixels

Conv Layer :

nn.Conv2d(in\_channels=3, out\_channels=32, kernel\_size=3, padding=1)

in\_channels == C

out\_channels = number of filters applied to the image in current layer(number of feature maps to be learnt)

padding = kernel\_size / 2

Output of a convolution layer for all batch\_size is:

(batch\_size,#feature\_maps,H’,W’)

H’ = (HeightImage – Kernel\_Size + 2\*Stride) / Padding + 1

Feature Maps – applying filters of C[in] x K x K for each element of the current image computing a dot product, use stride and also border your image using padding (zero\_padding)

Max pooling – downsampling a feature map for keeping the most important features only (slide a 2x2 or 3x3 window trough each element of the feature map and pick the max from it) maybe also use a stride

Batch Normalization: Problems when batchNorm was not discovered:

-vanishing gradients(gradienti mici care nu se actualizeaza eficient)

-exploding gradients(gradienti foarte mari destabilizand antrenarea)

Normalizing method applied to **feature maps**

L8

CNN architectures champions for ImageNet competition

AlexNet

VGG

GoogLeNet

ResNet (important)

-skip connections - Allow direct information flow between layers, solving vanishing gradient and accuracy degradation issues in deep networks.

-batch normalizaiton - Normalizes layer activations, stabilizing training and enabling higher learning rates

-standard block - Uses two 3x3 convolution layers without dimensionality reduction and is employed in shallower networks (e.g., ResNet-18 and ResNet-34)

-bottlle neck block - Uses a 1x1, 3x3, and 1x1 convolution structure to reduce parameters and computations, used in very deep networks (e.g., ResNet-50, ResNet-101).

Receptive field?!

Hello, iti las aici o intrebare: Am 2 layere de convolutie consecutive cu un kernel size de 3x3 sunt aproape echivalente cu un layer de 5x5 in sensul ca acopera aceasi zona din imaginea originala? iar numarul de parametri care trebuie invatati sunt mai putini pentru layerele cu kernel size 3x3, Am auzit si termenul ca au acelasi "receptive field", dar nu gasesc o explicatie intuitiva pentru mine. Partea cu eficienta computationala am inteles-o.

L9

CPU vs GPU

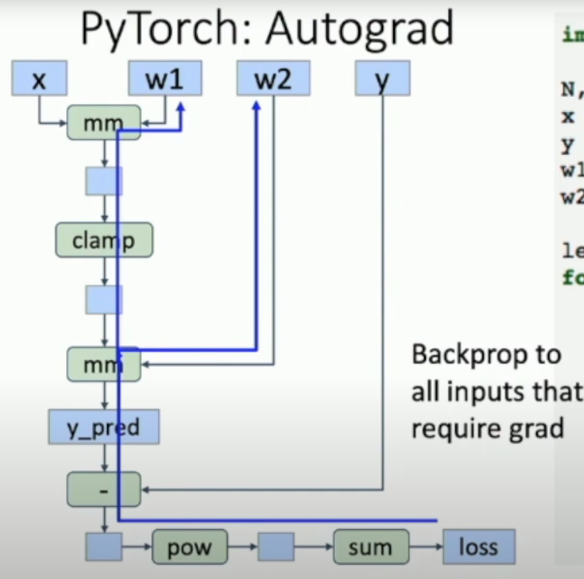
CPU Clock Speed in GHz – How many instructions per second can the CPU perform.

GPU(tiny mini computer) – has fans, memory modules, processor

Processor - has multiple “Streaming multiprocessors”

-in a Streaming Multiprocesor there are multiple Tensor cores that can multiply 4x4 (in 2019) matrices in one clock cylce

**Autograd**:

A computer screen shot of a computer code

AI-generated content may be incorrect.

**SET GRADIENTS TO ZERO – COMMON BUG**

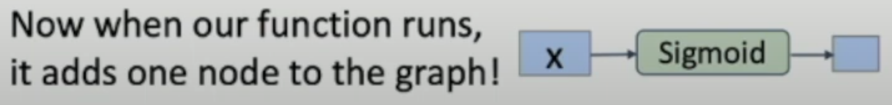
For each operation on tensor check if a tensor In an operation has requires\_grad=True if so build up to the computational graph so PyTorch can auto compute gradients

“with torch.no\_grad()” – tells pyTorch not to build a graph for its operation

A screenshot of a computer program

AI-generated content may be incorrect.

When you write your own python function be careful that in the computational graph it will add each operation

A person standing in front of a computer screen

AI-generated content may be incorrect.

You can visualize computational graph with different tools!

Torchviz/TensorBoard

A close-up of a computer screen

AI-generated content may be incorrect.