Introductory Workshop in Computer Vision

February 6, 2021
Naman Gupta

Overview

Prerequisites -

- Basics of Machine Learning - Regression, Classification, Gradient Descent
- Vanilla Neural Networks
- Familiarity with Python
- Enthusiasm to explore stuff yourself

What will we cover?

- Basics of Deep Neural Networks
- Handling Datasets
- Debugging DL models - Measuring model performance
- Leveraging State of the Art research
- Transfer Learning
- Model Ensembling

Introduction

- What is Computer Vision?

  CV is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

- Why do we need it?

  Automation - Human Resource is expensive

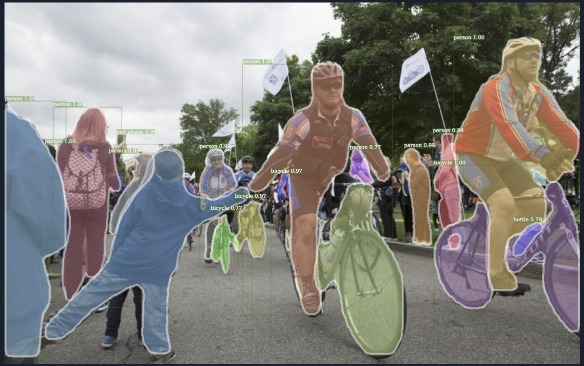
- Challenges?

  Complexity of the visual world

Introduction

A non-exhaustive list of tasks solved using Computer Vision and Deep Learning -

- Image Classification
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Human Pose Estimation
- Super Resolution
- Image Generation
- Image Captioning



Object Detection + Instance Segmentation

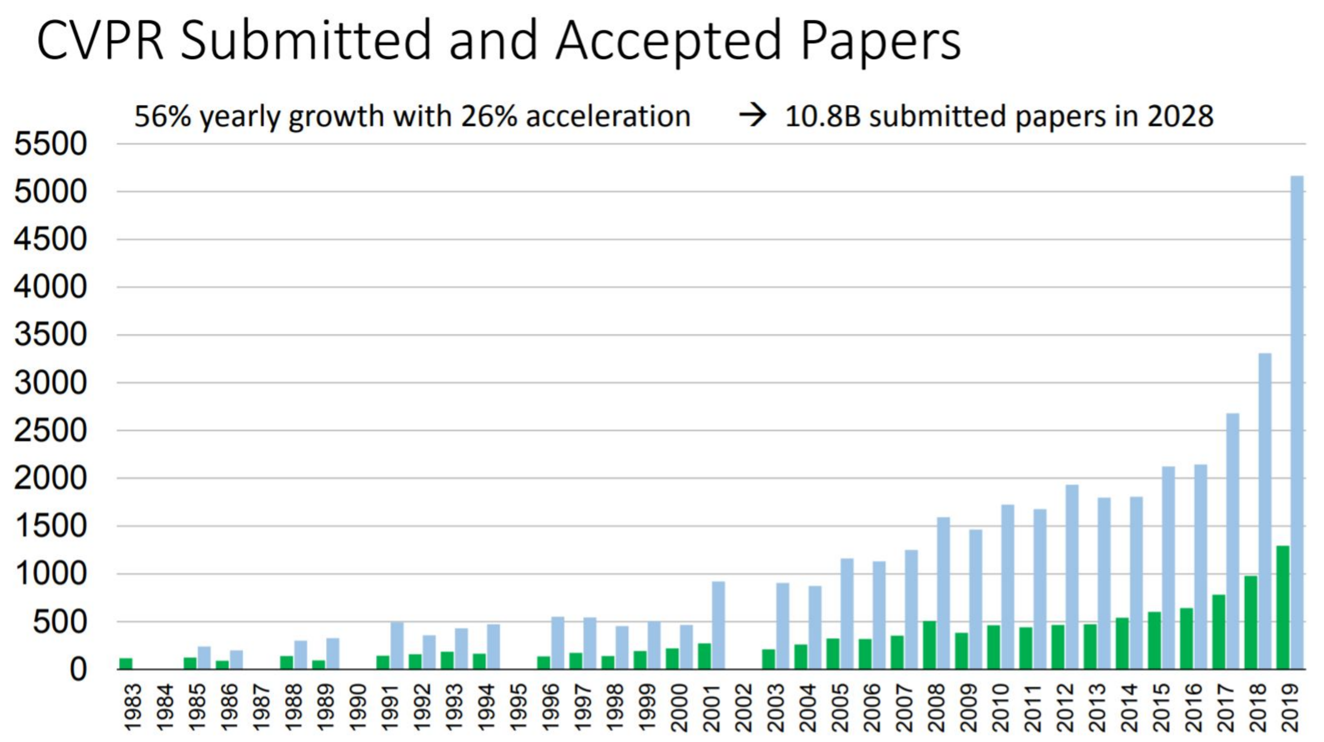
Introduction

Human Pose Estimation

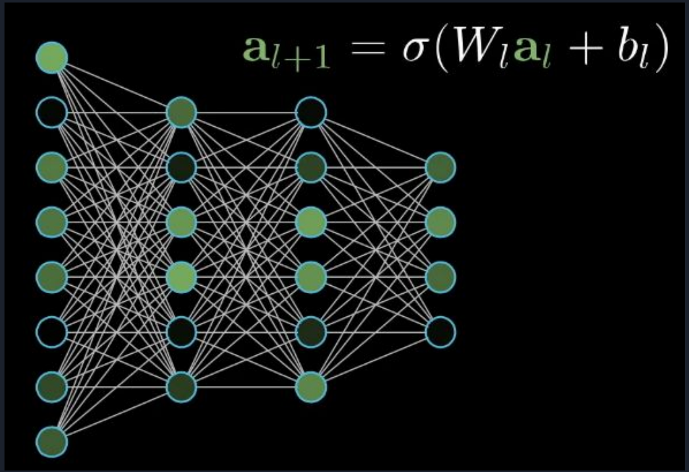
Image Generation
Faces generated by StyleGAN2

thispersondoesnotexist.com

Introduction - History



Refresher - Neural Networks



Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad \text{(BP1)}$$

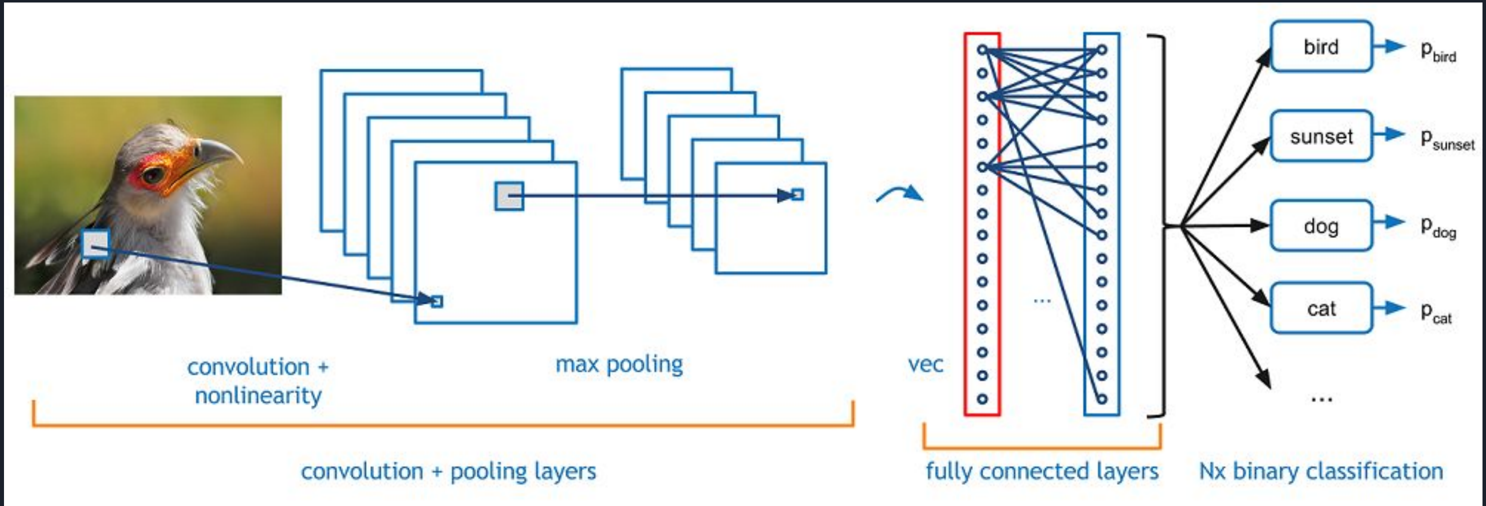
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \qquad \text{(BP2)}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \qquad \text{(BP3)}$$

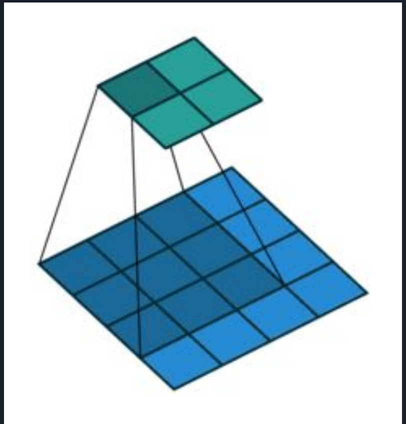
$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \qquad \text{(BP4)}$$

Resources:
http://neuralnetworksanddeeplearning.com/
[Neural networks - 3Blue1Brown]()

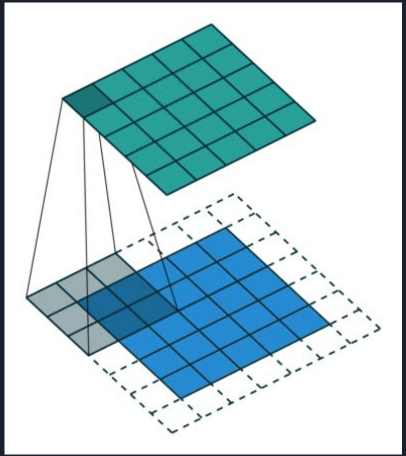
Neural Networks - Layers



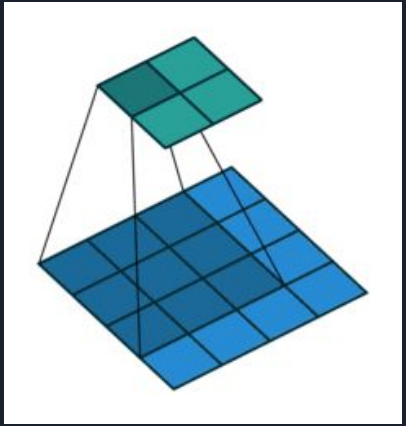
Neural Networks - Convolutions



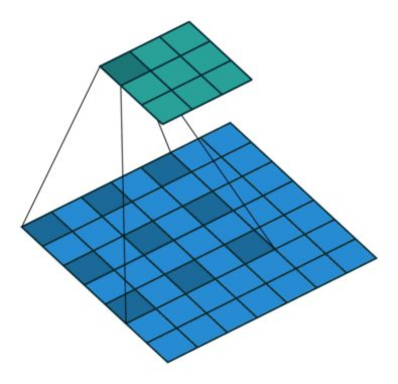
2D Convolution
Zero padding



2D Convolution
Same Padding



Transposed 2D Convolution

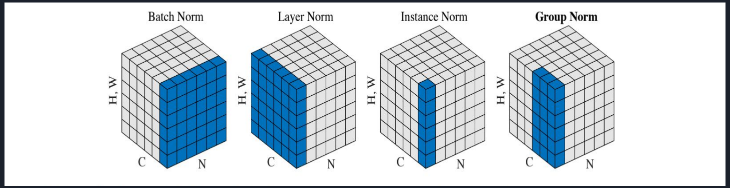


Dilated Convolution

https://theano-pymc.readthedocs.io/en/latest/tutorial/conv_arithmetic.html

Neural Networks - Layers

- Normalization - Reducing internal variance
  - Batch Norm
  - Layer Norm
  - Group Norm
  - Instance Norm
- Dropout
- SoftMax
- MaxPool vs Strided Convolutions
- Transposed Convolutions
- Dilated Convolutions



Neural Networks - Activation Functions

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- Hardswish
- Many others...

What should I use?
There’s no correct answer. Avoid Sigmoid and Tanh unless necessary. Why?
Go with ReLU or one of its variants.

Neural Networks - Loss Functions

Depends on the task at hand.

- Classification : Cross Entropy Loss, Focal Loss
- Regression : L2 (Mean Squared Error), L1, Smooth L1
- Detection : Multi Objective Loss - mIoU + CLS + REG
- Semantic Segmentation : Cross Entropy, Dice Loss

General Tips -

- Check if your dataset is imbalanced
- For multi objective loss functions it is important to tune the relative weightage of individual losses
- Loss is not a metric (More on this later)

Neural Networks - Optimizers

- SGD
- RMSProp
- Adam
- RAdam
- AdamW
- LookAhead

Is one optimizer better than other?

Datasets

- Why do DNN require a lot of data? Is there a way out?
- Exploratory Data Analysis - Why is it important?
- Dataset Splits - Train, Validation and Test
  - Why is it needed and what’s the correct way?
  - k-fold Cross Validation
- How to handle imbalanced data?
  - Oversampling
  - Undersampling
  - SMOTE
  - Using weighted loss functions
  - Getting more data!

Datasets - Augmentation

What type of augmentation do I use? Depends on the specific use case.

Some general augmentation techniques -

- Horizontal / Vertical Flip
- Shift Scale Rotate
- Noise , Blur
- Brightness / Contrast , Hue Saturation

Some other non-trivial augmentation techniques -

- CutMix, Mixup, Mosaic, CutOut
- AutoAugment

Can GANs help in augmentation?

Debugging Models - Tracking Loss

- Is training loss sufficient?
- Interpreting loss across epochs
  - Learning Rate
    - Warmup
    - Decay - Cosine Annealing, Step Scheduler
  - Underfitting / Overfitting
- Measuring a model’s performance - Loss vs Metric

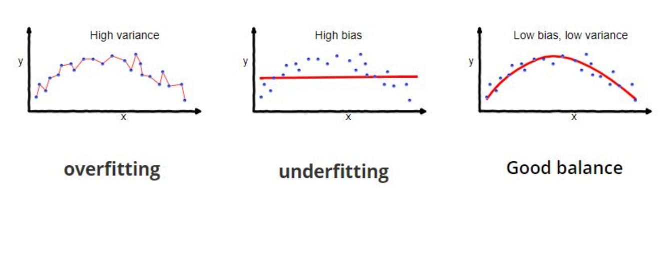
Debugging Models - Bias vs Variance

High Bias - Underfitting

- Decrease Regularization
- Use larger models (Naively adding more layers doesn't always work !)

High Variance - Overfitting

- Increase Regularization
  - Weight Decay - L2
  - Dropout
  - BatchNorm! - has regularizing properties
- Early Stopping
- Data Augmentation
- Get more data!

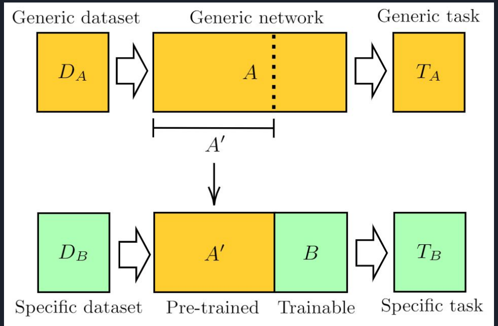


Metrics are important!

- Classification - Accuracy isn’t a good measure
  - Confusion Matrix - Precision, Recall and F1 Score
  - AUC Curve - TPR vs FPR
- Detection
  - Mean Average Precision (mAP) : Area under the PR curve
- Segmentation
  - Mean Intersection over Union (mIoU)
  - Dice coefficient
- GANs
  - Frechet Inception Distance (FID)
- Super Resolution
  - Peak Signal to Noise Ratio (PSNR)

Transfer Learning

- Pretrained weights are generally better than random initialization.
- How do you use a pre-trained model?
  - Replace the head with a randomly initialized one.
  - Freeze parameters in initial layers
    How many layers to freeze?
  - Or even train the whole model with your dataset
    Will converge faster than a randomly initialized model
- Examples?
  - Use any pre-trained model on ImageNet, finetune on your specific dataset
  - StyleGAN2-ADA : Train a GAN with your own data with less than 1000 examples!



SoTA Review - Classification

- Benchmark Datasets
  - ImageNet - One dataset to rule them all !
- Popular Breakthroughs
  - AlexNet - Deep Learning Era begins *2012*
  - VGG-19 - Neural Networks go deeper *2014*
  - ResNet - Skip Connections, More than 100 Layers *2015*
  - MobileNetV2 - Inverted Residuals *2018*
  - EfficientNet - AutoML and Model Scaling *2019*
  - ViT, DeiT - Transformers in Computer Vision *2020*

Do you need these complex architectures for your two class dataset? No.

SoTA Review - Detection

- Benchmark Datasets
  - MS COCO
  - Pascal VOC
- Popular Breakthroughs
  - FasterRCNN *2016*
  - MaskRCNN *2017*
  - YOLO *2016-20*
  - EfficientDet *2020*

SoTA Review - Semantic Segmentation

- Benchmark Datasets
  - Cityscapes
  - Pascal VOC
- Popular Breakthroughs
  - DeepLab *2016-18*
  - UNet - Medical Images *2015-18*
  - PSPNet *2017*

Ensembling and TTA

Ensembling

- Training multiple models, combining their predictions
  - Simple Mean / Weighted Mean
- Should be used as a last resort to squeeze that last percent of accuracy

Test Time Augmentation (TTA)

- Augmenting the test image to get multiple images, passing each augmented image to the model and combining all the predictions

Some Practical Tips for the Hackathon

- Do a thorough Data Analysis before going into modelling
  - In case of imbalanced datasets use weighted loss functions
    Example - Focal Loss / Weighted Cross Entropy for Classification
  - Check if there are inherent labelling errors in the data
- Always use a cross validation set
- Don’t use complex models blindly: With great models comes great responsibility!
  - Example - Using EfficientNet for MNIST
- Leverage transfer learning and SoTA models as much as possible, but don’t push too hard for latest research.
- Look for open source implementations before getting into your own code implementation
- Don’t forget data augmentation!
- Look for related datasets if any - Google Dataset Search
- Use Ensembling and TTA if necessary

What happened in 2020?

Except Covid-19

- Transformers in Computer Vision
  - Vision Transformer achieves SoTA in ImageNet
  - DeiT
  - OpenAI - CLIP : Connecting Images with Text
  - OpenAI - Dall-E : Generating Images from Text
- YOLOv4
- EfficientDet
- ResNeSt

PyTorch example

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

```python
for epoch in range(2):  # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:    # print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

https://pytorch.org/tutorials/

Conclusion

Don't get intimidated! - A lot was covered in the previous slides and you do not need to know everything.
Deep Learning as a field is progressing every day at a very fast pace. Hence keeping track of everything is not practically possible neither needed. Nevertheless keep exploring, we all are learning together!

Additional Resources

- Deep Learning Specialization - Coursera
- CS231n by Stanford University - Lectures available on YouTube
- Kaggle - Competitions and Datasets
- paperswithcode.com - Search SoTA architectures and their code implementations
- arxiv-sanity.com - See what’s trending in research

Awesome repositories related to Computer Vision

- https://github.com/rwightman/pytorch-image-models
- https://github.com/albumentations-team/albumentations/
- https://github.com/rwightman/efficientdet-pytorch
- https://github.com/pytorch/vision

Contact Me

Naman Gupta
Facebook - naman.iitk
Email - namangup@iitk.ac.in

Questions?