Supervised Learning

List of various loss and activation functions, derivatives (not complete)

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - w^\top x - b)^2 \qquad \mathcal{L}_{BCE} = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \frac{d\sigma(x)}{dx} = \sigma(x) \times (1-\sigma(x))$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad \frac{d\tanh(x)}{dx} = 1 - \tanh^2(x)$$

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \qquad \frac{d\text{Softmax}(x_i)}{dx_j} = \begin{cases}\text{Softmax}(x_i) \times (1-\text{Softmax}(x_i)) & \text{if } i = j \\ -\text{Softmax}(x_i) \times \text{Softmax}(x_j) & \text{if } i \neq j\end{cases}$$

Performance Evaluation

- Using entire data for training can lead to overfitting, and true error rate is underestimated. Therefore, reserve some data for validation, so that we can tune the hyperparameters.
- Validation methods - **Holdout**(splitting into traning and validation),**K-Fold Cross-validation**(use K-1 folds for training and 1 fold for validation)

Debugging ML Models

- Large gap between train and test error means **overfitting**. It could also mean the model needs to be trained on more data.
- Small gap between train and test error means **underfitting**. It could also mean the model needs to be more complex.
- **Ablative Analysis**: Tries to explain the difference between some baseline performance (usually poorer) and current performance. Solution is to reduce complexity of the model by removing features one by one and see till we reach baseline performance.

Detour: Information Theory

- $I(x) = -\log P(x)$, $H(X) = \mathbb{E}_{x\sim P}[I(x)] = -\mathbb{E}_{x\sim P}[\log P(x)]$

$$D_{KL}(P||Q) = \mathbb{E}_{x\sim P}\left[\log\frac{P(x)}{Q(x)}\right] = \mathbb{E}_{x\sim P}[\log(P(x)) - \log(Q(x))] = H(P,Q) - H(P)$$

Visual Recognition

Convolutional Neural Networks

- **Convolutional Layer** : Used for feature extraction. It has a set of learnable filters, which are convolved with the input. The filters are learned using backpropagation.
- **Pooling Layer** : Used for downsampling. It reduces the spatial dimensions of the input. It also reduces the number of parameters and computations in the network, and hence controls overfitting.
- Hidden Unit Channel = Map = Feature = Feature type = Dimension
- Weights for each channel = kernels = filters
- **Size of next layer** = $\frac{W-F+2P}{S} + 1$ where W is the input size, F is the filter size, P is the padding, and S is the stride.

Semantic Segmentation using Fully Convolutional Networks

- **Semantic Segmentation**: Determine what is in the image and where it is. It assigns a class to each pixel in the image.
- Higher layers of typical CNNs are nonspacial, but by making them spacial, we can both what and where. This is done by using fully convolutional networks(replacing FC layers with convolutional layers).
- **Learnable Upsampling**:
  - Nearest Neighbour: Simply repeat the pixels.
  - Bilinear Interpolation: Take a weighted average of the 4 nearest pixels.
  - FCNs: Learnable kernel function that defines the map for interpolation.
- **Transposed Convolution: Fractional Stride**
  - Normal Stride 1: You take each pixel of your input image, you multiply each pixel of your say 3x3 kernel with the input pixel to get a 3x3 weighted kernel, then you insert this in your output image. Where the outputs overlap you sum them.
  - You can also use a stride larger than one to increase the upsampling effect (or some would call this a convolution with a fractional stride 1/f)
- **Skip Connections**: Skip some of the connections to the next layer, and connect to a layer further down the network. This helps in preserving the spatial information. Combining fine layers and coarse layers lets the model make local predictions that respect global structure
- **Encoder-Decoder Architecture**: Encoder is a typical CNN, and decoder is a FCN. The encoder is used to extract features, and the decoder is used to upsample the features to the original size of the image.

- **Altrous Convolution** (DeepLabV3): Convolution with a kernel that has holes in it. It is used to increase the receptive field of the network without increasing the number of parameters.
- U-Net copies feature maps from encoder to serve as cues for the decoder
- SegNet uses pooling and non-learnable upsampling followed by conv layers in the decoder

Object Detection

- **Feature Maps** = features and their locations. The problem is how to predict location and class of multiple objects in an image.
- Applying CNN to many different crop images is not efficient. Solution is to use Region Proposals(regions which are likely to contain objects) and then apply CNN to these regions. This idea is called **Regional CNN**(R-CNN).
- Regional Proposal $\rightarrow$ Resize to standard size $\rightarrow$ CNN $\rightarrow$ Bounding Box Regression and SVM.
- Fixed-length feature vector is required for the SVM. To create fixed length features from a feature map, we use **Spacial Pyramid Matching**.
- **Fast R-CNN**: Instead of applying CNN to each region proposal, apply CNN to the entire image, and then use the feature map to extract features for each region proposal. Can be done using SPP or RoI pooling.
- **Faster R-CNN**: Use a Region Proposal Network to generate region proposals. The RPN is a fully convolutional network that takes the feature map as input and outputs region proposals.
- **RPN**(Region Proposal Network): Sliding window over the feature map, and for each window, predict the probability of an object being present and the bounding box of the object.

YOLO(You Only Look Once)

- Divide the image into a grid, and for each grid cell, predict the bounding box and the class of the object. Then combine the box and predictions. Finally do Non-Maximum Suppression to remove duplicate boxes.
- During training, match example to the right grid cell, adjust that cell's class prediction, adjust the bounding box prediction, and adjust the confidence score, decrease confidence score for other boxes.
- For cells which don't have a ground truth object, decrease the confidence score. Don't adjust the class probabilities or bounding box prediction.
- Using MSE as a loss function is not a good idea since loss for larger objects is larger, and smaller for smaller boxes. Instead we use Loss functions like IoU(Intersection over Union) or GIoU(Generalized IoU).
- **IoU** = $\frac{\text{Area of Overlap}}{\text{Area of Union}}$, **GIoU** = IoU - $\frac{\text{Area of Bounding Box}-\text{Area of Union}}{\text{Area of Bounding Box}}$, where Area of Bounding Box is the area of the smallest bounding box that contains both the predicted and ground truth bounding boxes.
- **DIoU** and **CIoU** are other loss functions that can be used. $DIoU = IoU - \frac{d^2}{c^2}$, d is the distance between the centers of the predicted and ground truth bounding boxes, and c is the diagonal of the smallest bounding box that contains both the predicted and ground truth bounding boxes.

Evaluation Metrics

- For Regression: MSE, MAE, RMSE
- For Classification: Accuracy, Precision, Recall, F1 Score
  - Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$, Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$, F1 Score = $\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$
- IoU is typically used to assess quality of locality.

Geometry

Camera Model

- **Pinhole Model**
  - The camera captures pencils of rays, all rays go through a single point, which is called the center of projection (focal point)
  - The image is formed on the image plane (film). 3D to 2D transformation occurs via a perspective projection.
- Information lost in perspective projection: Length and distances, Angles

Modelling Projections

- Use the pinhole model as an approximation. Put optical center(Centre of Projection) at the origin, and the image plane at $z = f$.
- The camera looks down +z axis, and y-axis points down

Geometric Transformations

- Translation: $T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, Scaling : $S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
- Scaling and Translation $\neq$ Translation and Scaling
- Rotation: $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$, θ is the angle of rotation in counter-clockwise direction. For any matrix to be a rotation matrix, it must be orthogonal, and have a determinant of 1.