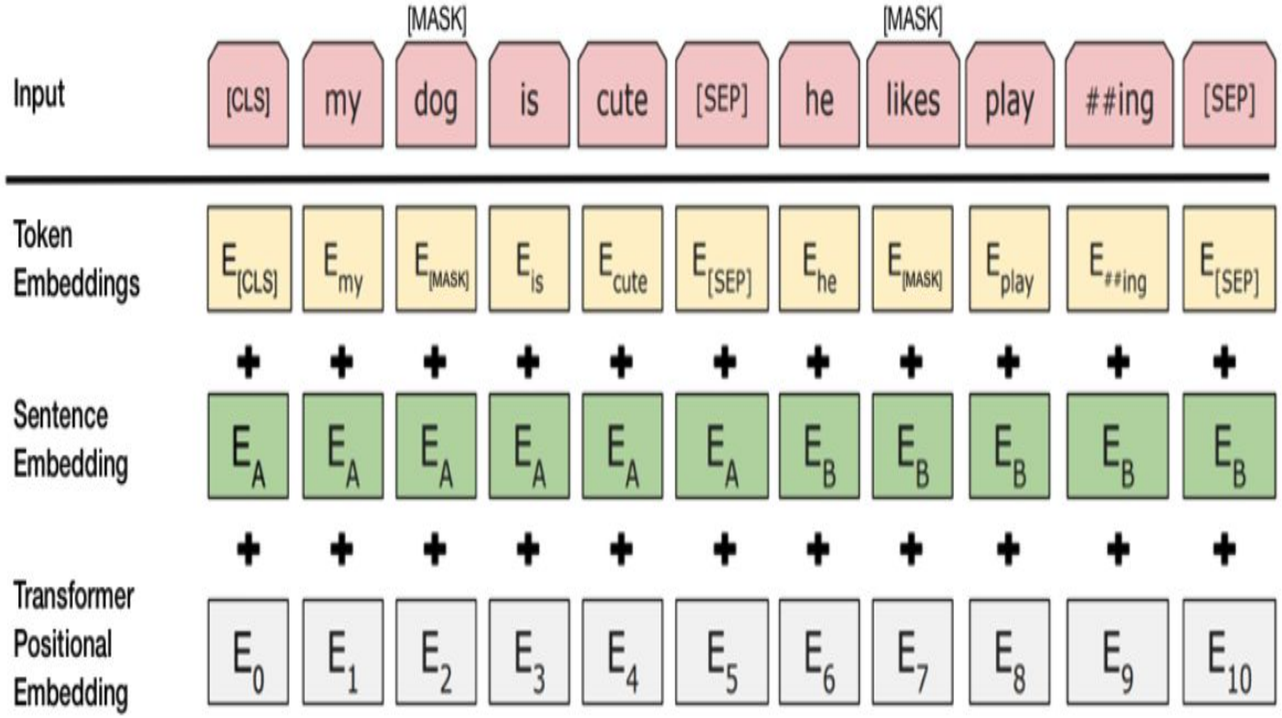
BERT(Bidirectional Encoder Representations from Transformers)

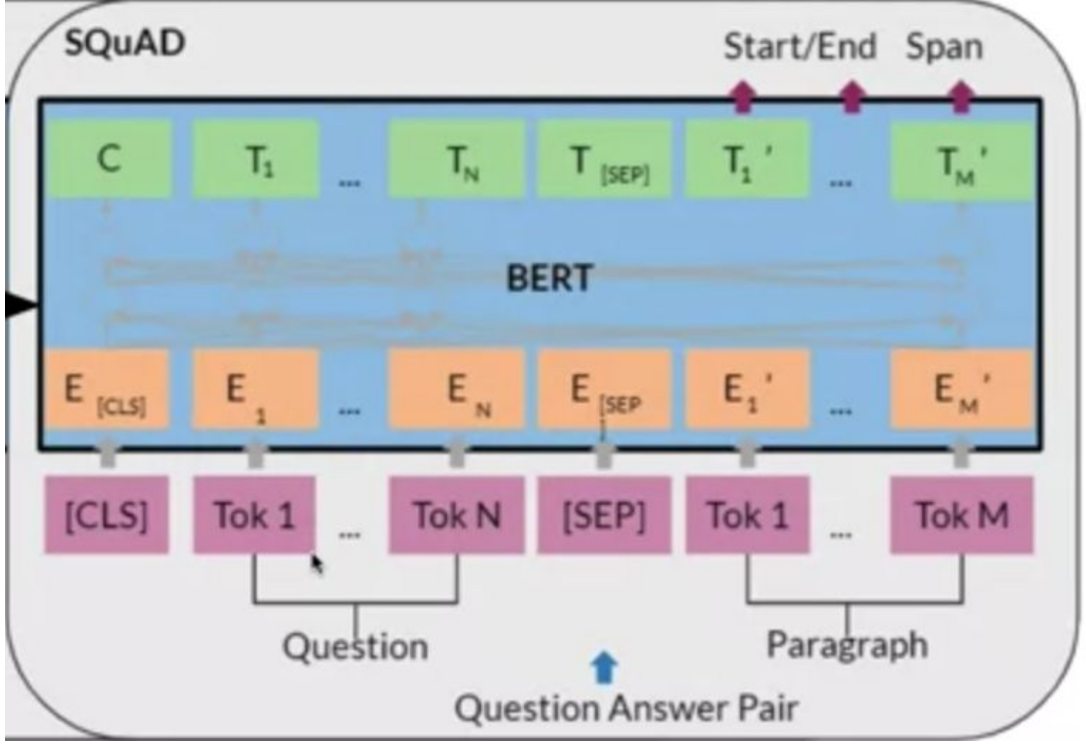
- Transformers (Attention Model)
- Deeply Bidirectional Training
- Masked Language Model and Next sentence prediction for training

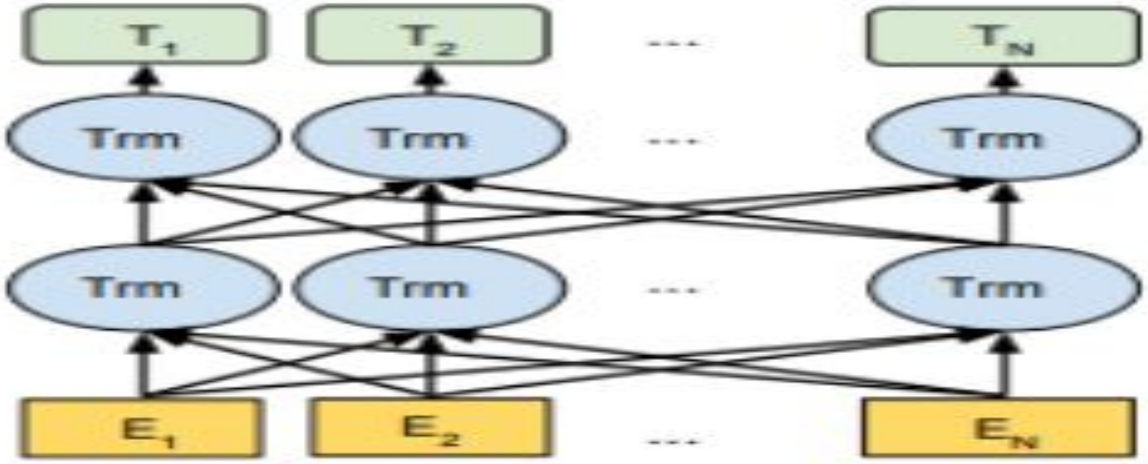
- Open AI GPT – right to left – transformers
- ELMo – LSTM – shallowly bidirectional

- NLP preprocessing

1. Remove html tags, special characters, numbers, stopwords
2. Lemmatization – Converting word to its base
3. Tokenization
4. Normalization – lowercase, whitespace

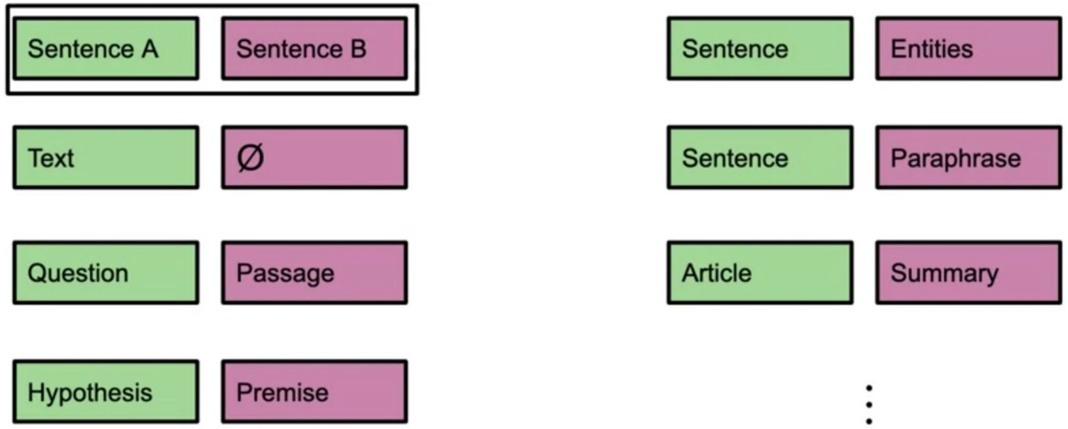




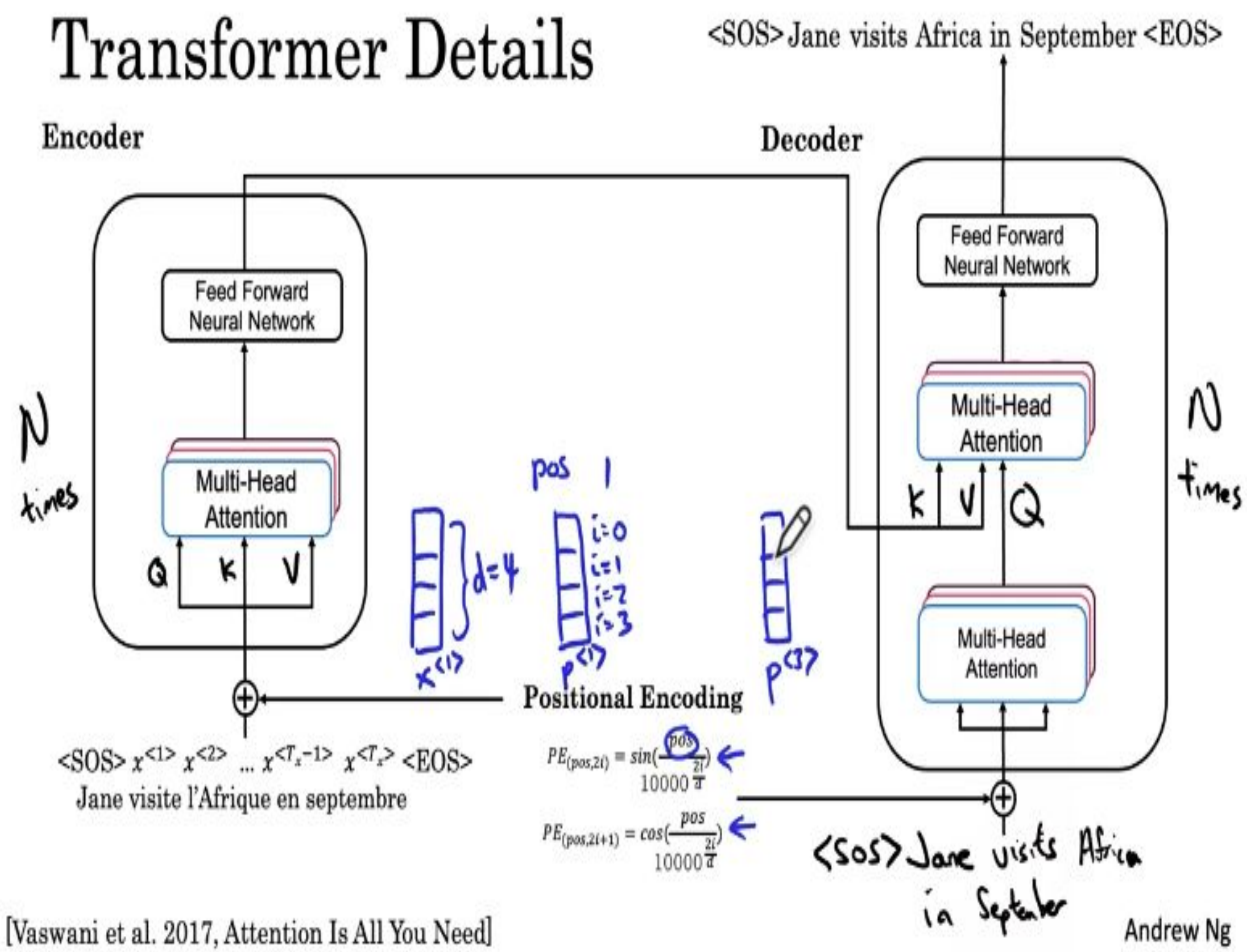


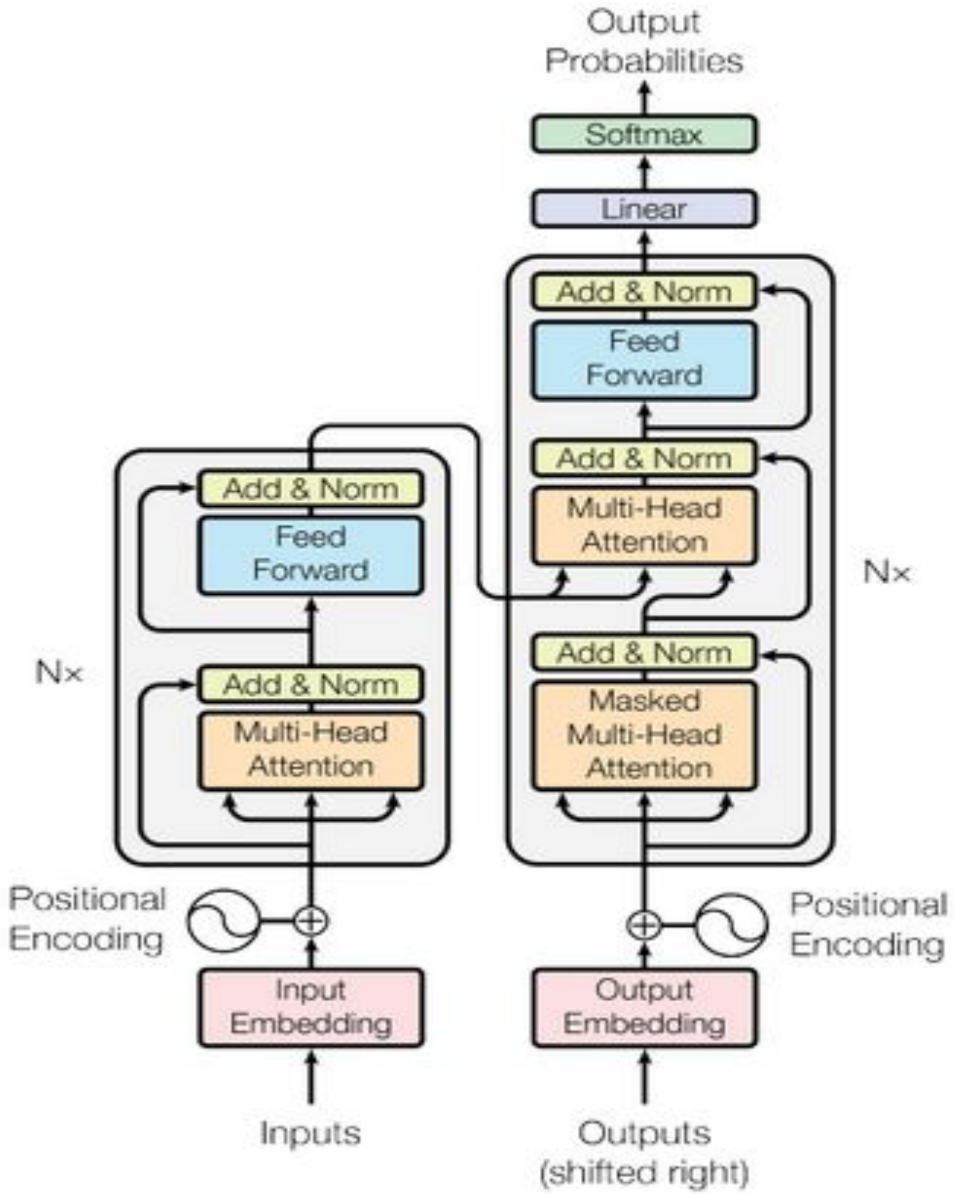
BERT Architecture

Summary

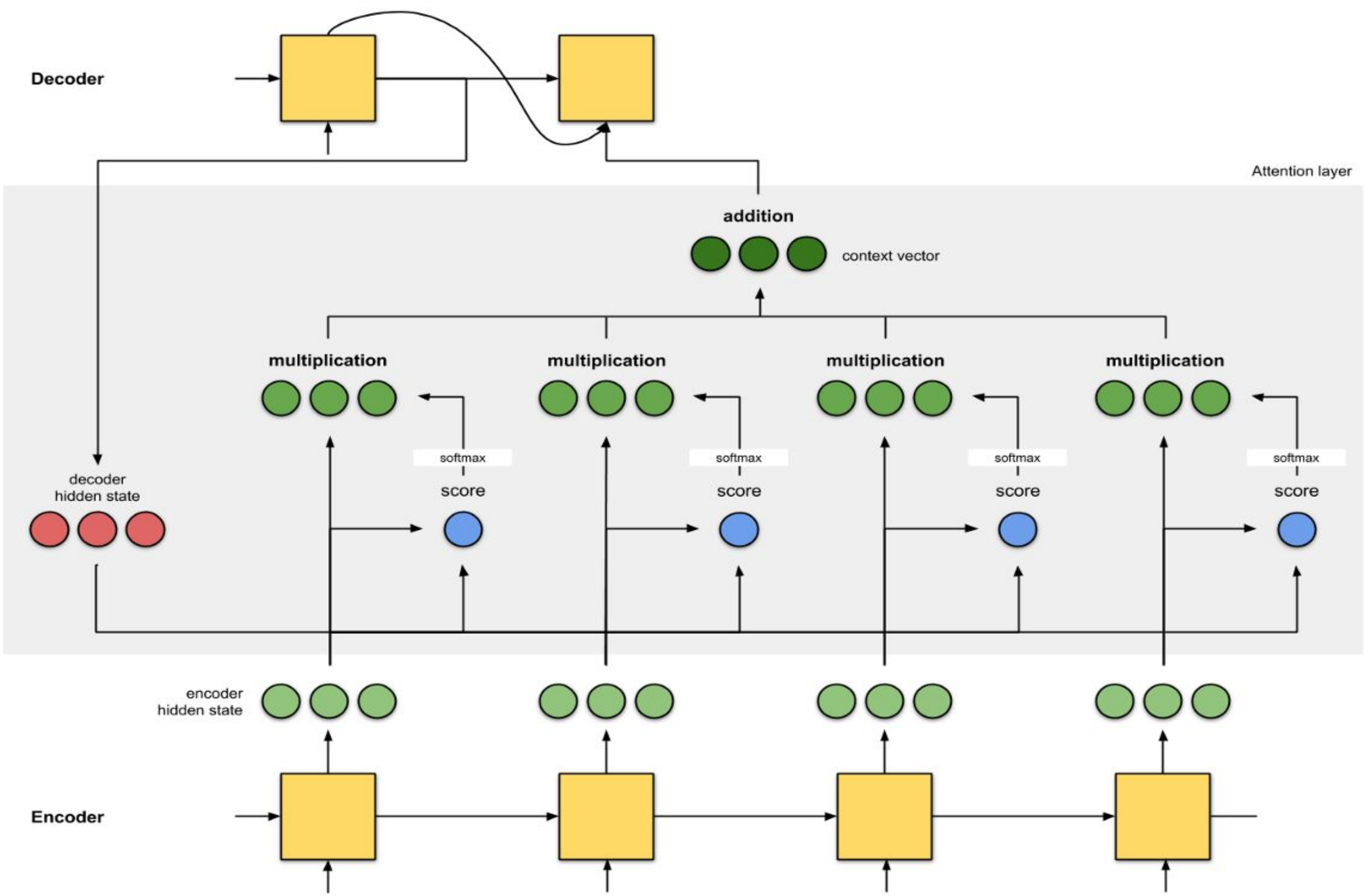


Transformers





Quadratic complexity for vision transformer – solved by swin and pyramid transformers



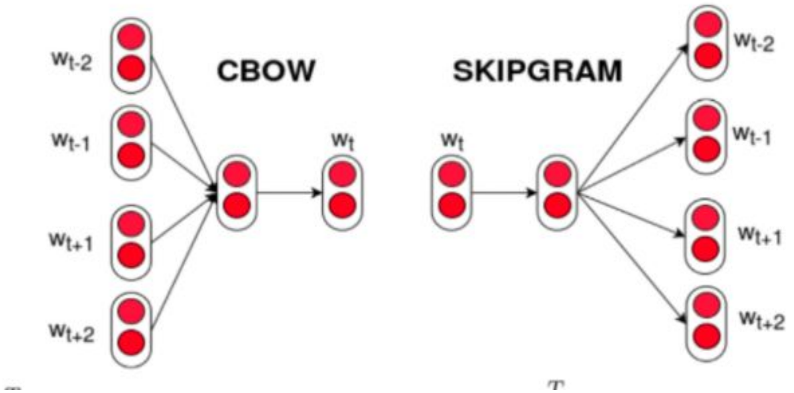
https://peltarion.com/blog/data-science/self-attention-video

Embeddings

- *Word2vec* embeddings are based on training a *shallow feedforward neural network* while *glove embeddings* are learnt based on *matrix factorization* techniques.
- Glove model is based on leveraging *global word to word co-occurance counts* leveraging the entire corpus. Word2vec on the other hand leverages *co-occurance within local context (neighbouring words).*

Word2Vec Training

The following figure pictorially shows these models.



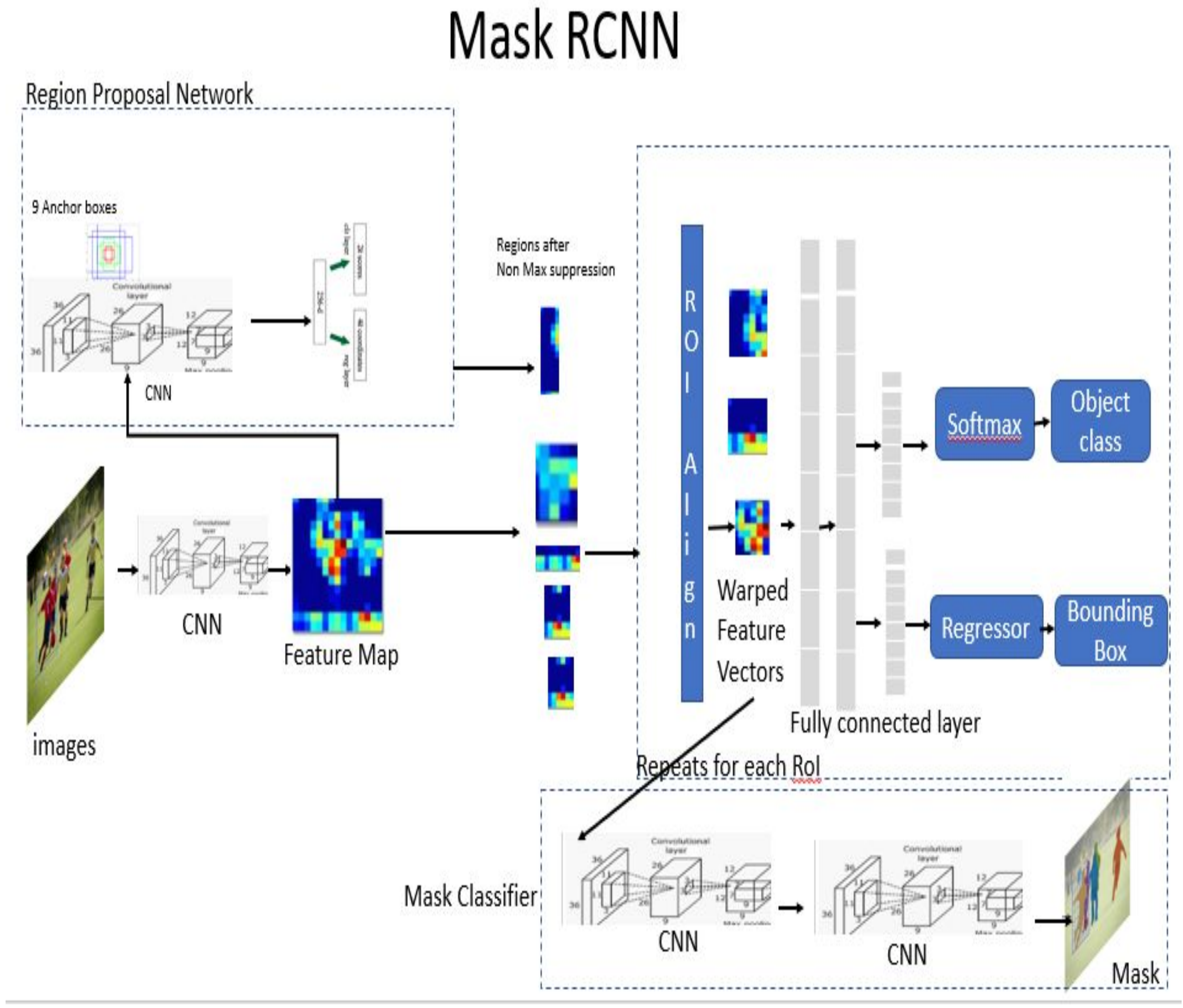
Glove embeddings Function

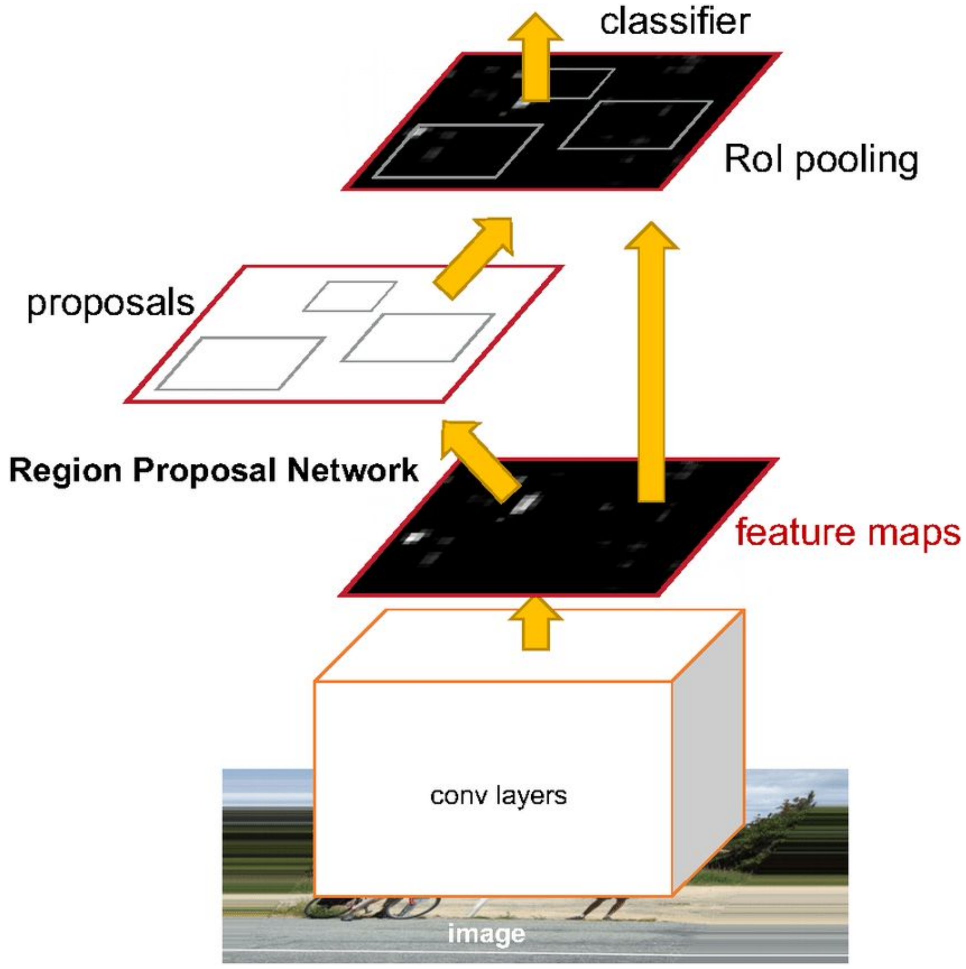
$$\text{minimize } \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i - b_j' - \log X_{ij})^2$$

Transformers – self attention

- **Challenges**
- Despite being so good at what it does, there are certain limitations of seq-2-seq models with attention:
- Dealing with long-range dependencies is still challenging
- The sequential nature of the model architecture prevents parallelization. These challenges are addressed by Google Brain's Transformer concept

Mask-rcnn and Faster-rcnn

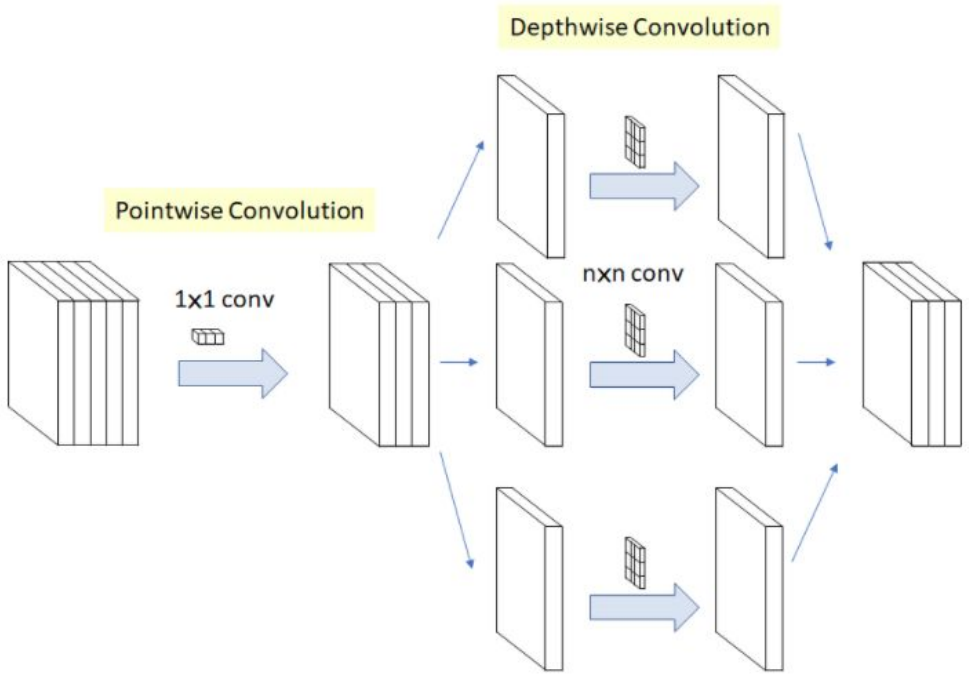




Exception

- Exception uses modified depth wise separable convolution, hence reduces the parameters i.e overfitting.
- Consists of residual connections as well.
- Depth wise separable convolution – each conv filter is applied across each channel, then pointwise convolution is used to down sample (In exception reverse).

2. Modified Depthwise Separable Convolution in Xception



The Modified Depthwise Separable Convolution used as an Inception Module in Xception, so called "extreme" version of Inception module (n=3 here)

Image localization

- Image segmentation

1. Mask-RCNN – similar to Faster-rcnn
2. U-Net - Upsampling (Transposed convolution)

- Object detection

1. SSD
2. YOLO
3. Faster-rcnn

Bounding Box Regression

Given a predicted bounding box coordinate $\mathbf{p} = (p_x, p_y, p_w, p_h)$ (center coordinate, width, height) and its corresponding ground truth box coordinates $\mathbf{g} = (g_x, g_y, g_w, g_h)$, the regressor is configured to learn scale-invariant transformation between two centers and log-scale transformation between widths and heights. All the transformation functions take $\mathbf{p}$ as input.

$$\begin{aligned}
\hat{g}_x &= p_w d_x(\mathbf{p}) + p_x \\
\hat{g}_y &= p_h d_y(\mathbf{p}) + p_y \\
\hat{g}_w &= p_w \exp(d_w(\mathbf{p})) \\
\hat{g}_h &= p_h \exp(d_h(\mathbf{p}))
\end{aligned}$$

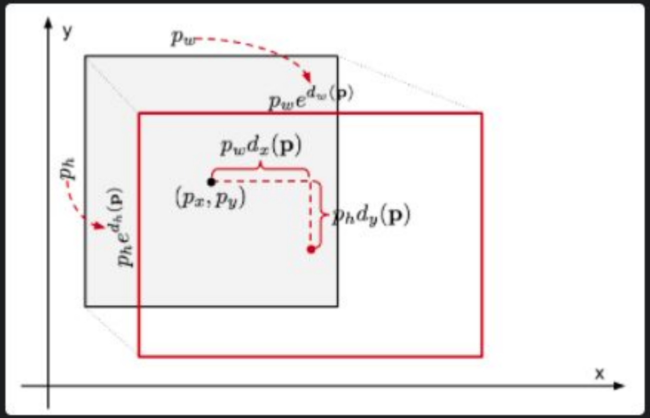


Fig. 2. Illustration of transformation between predicted and ground truth bounding boxes.

An obvious benefit of applying such transformation is that all the bounding box correction functions, $d_i(\mathbf{p})$ where $i \in \{x, y, w, h\}$, can take any value between [-∞, +∞]. The targets for them to learn are:

$$\begin{aligned}
t_x &= (g_x - p_x)/p_w \\
t_y &= (g_y - p_y)/p_h \\
t_w &= \log(g_w/p_w) \\
t_h &= \log(g_h/p_h)
\end{aligned}$$

A standard regression model can solve the problem by minimizing the SSE loss with regularization:

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$

Improving performance of deep learning

- Data Augmentation
- Class Imbalance
- Fine tuning of hyper parameters (learning rate, regularization parameter, Batch size)
- Model Ensembles
- Grid Search for hyperparameters
- Dropout(overfitting)
- Batch normalization
- Shuffling dataset
- Reducing Complex network (overfitting)

you can use this formula `[(W−K+2P)/S]+1`.

- W is the input volume - in your case 128
- K is the Kernel size - in your case 5
- P is the padding - in your case 0 i believe
- S is the stride - which you have not provided.

TF-IDF and similarity measures

- TF = how frequently a term occurs in a document
- IDF = Log (Total # of Docs / # of Docs with the term in it)

word	TF				IDF	TF * IDF			
	d1	d2	d3	d4		d1	d2	d3	d4
Italian	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
Restaurant	1/8	1/6	1/6	1/6	log(4/4)=0	0	0	0	0
enjoy	1/8	1/6	1/6	0/6	log(4/3)=0.13	0.016	0.02	0.02	0
the	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
best	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
pasta	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
American	0/8	1/6	0/6	1/6	log(4/2)=0.3	0	0.05	0	0.05
hamburger	0/8	1/6	0/6	0/6	log(4/1)=0.6	0	0.1	0	0
Korean	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0
bibimbap	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0

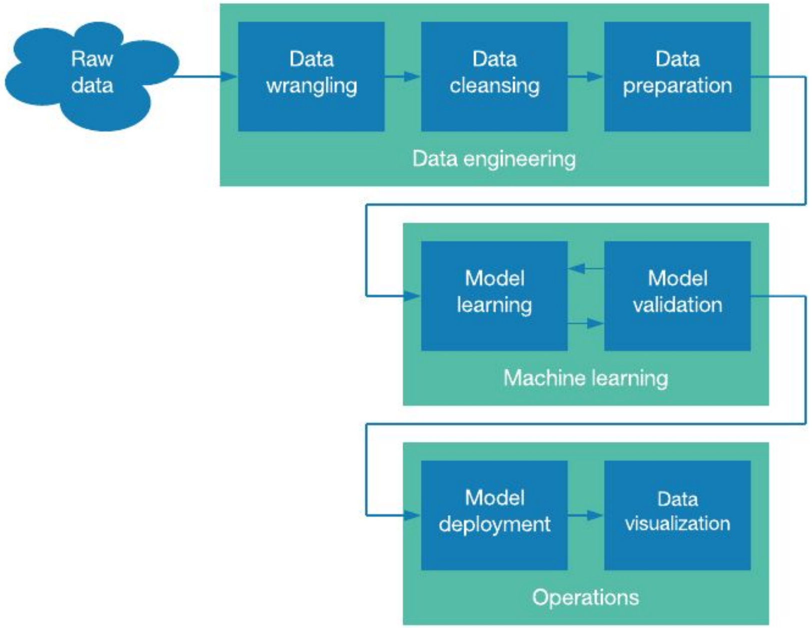
Cosine Similarity

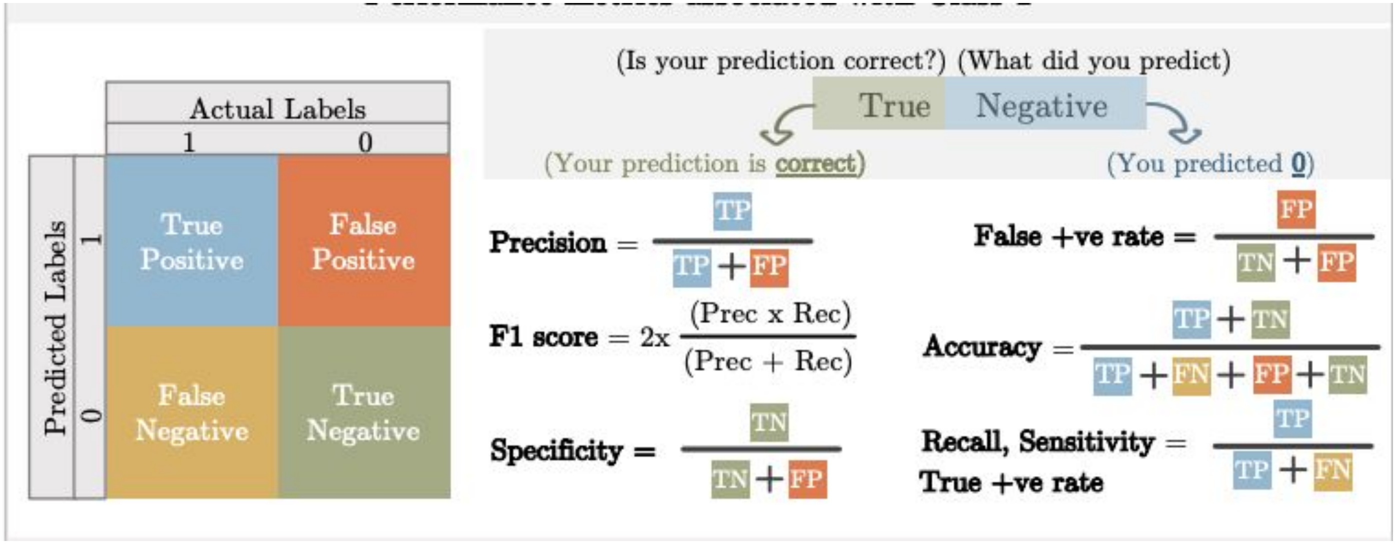
$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Jaccard Similarity J (A,B) = | Intersection (A,B) | / | Union (A,B) |

Accuracy: %age correct prediction
Precision: Exactness of model

Recall: Completeness of model
F1 Score: Combines Precision/Recall





$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

Accenture Project

- Purpose/situation: Build an e-commerce like application for generation of Case Report Forms (CRF).
- Task/Steps involved – **Extraction** of relevant data from protocol document; **Recommending** similar clinical studies, **providing** functionalities for selection, editing, filtering and aggregation of Forms and Fields;
- Action: Programmed for extraction using python-docx, lxml; Proposed and developed ranking algorithm for recommendation; designed and developed database and Backend API's for selection, searching, filtering and aggregation.
- Result: reduced the effort from 17 weeks to about 2-4 weeks.

Weeds3d

- Creating annotated data for semantic segmentation and biomass prediction from RGB-D images of field.
- Development of data pipelines for storage of the images and annotations.
- Data is collected using OAK-D camera, hence creation of data pipelines using python and DepthAI library. Annotations will be stored using MongoDB.
- As a result, insights will be provided to the farmer for efficient usage of herbicides.

Terrain identification from time series sensor data

- Terrain identification and activity recognition from time series sensor data for assisting prosthetic limbs. Labeled data is collected for various subjects for (0) standing or walking on solid ground, (1) indicates going down the stairs, (2) indicates going up the stairs, and (3) indicates walking on grass.
- Data preparation. Modeling. Evaluation.
- Based on literature for activity recognition from sensor data, trained ensemble model of 1D-CNN and LSTM. (As a part of this experimented with different input structures, various window sizes and hyperparameter tuning). Chose the best model based on F1-Score on validation split.
- Achieved 0.87 F1-score on the hidden test and was in the top-5 best performing models in the class.

Foraminifera species identification

- Foraminifera species identification requires domain expertise and is crucial for oceanographic studies. Motive of the project is to solve the problem of fine-grained image classification and data imbalance.
- Classification of 35 species from 35000 imbalanced dataset of images.
- Trained CNN models like VGG and Exception for establishing a baseline. Achieved higher accuracy by training bilinear CNN models of the same. Increased accuracy by hyperparameter tuning and using class weights for a modified loss function.
- pooled outer product of features derived from two CNNs and capture localized feature interactions in a translationally invariant manner. Successful in capturing highly localized features.
- Achieved better accuracy than the reference paper. And also received maximum marks for the project and the course.
- (VGG – extension of alexnet with 3x3 conv filters instead of 11x11).

Automatic number plate detection and identification

- Built an API for retrieving the license plate number from an image.
- Annotation. Model training. Evaluation. OCR. Integration.
- Tried various open source classical CV solutions. Annotated the images using labelImg for bounding boxes. Tried open source end-to-end pipelines (data not sufficient). Leveraged transfer learning of Faster-RCNN and YOLO for detection of number plate. Extended the module to open source OCR tool for license plate number identification. Integrated the module with a chatbot using Flask API.
- Successfully integrated the module to a chatbot for an insurance based product.

Active Learning -

- For selecting subset of data for labeling
- The steps to use active learning on an unlabeled data set are:
  - The first thing which needs to happen is that a very small subsample of this data needs to be manually labeled.
  - Once there is a small amount of labeled data, the model needs to be trained on it. The model is of course not going to be great but will help us get some insight on which areas of the parameter space need to be labeled first to improve it.
  - After the model is trained, the model is used to predict the class of each remaining unlabelled data point.
  - A score is chosen on each unlabeled data point based on the prediction of the model.
  - Once the best approach has been chosen to prioritize the labeling, this process can be iteratively repeated: a new model can be trained on a new labeled data set, which has been labeled based on the priority score. Once the new model has been trained on the subset of data, the unlabeled data points can be run through the model to update the prioritization scores to continue labeling. In this way, one can keep optimizing the labeling strategy as the models become better and better.
- Prioritization scoring mechanism - least confidence, margin sampling (difference between probability of 1st maximum and 2nd maximum), entropy.

Algorithm 1 BADGE: Batch Active learning by Diverse Gradient Embeddings

Require: Neural network $f(x;\theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of examples in a batch B.

Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with queried labels.
Train an initial model θ_1 on S by minimizing $\mathbb{E}_S[\ell_{\mathrm{CE}}(f(x;\theta), y)]$.
for $t = 1, 2, \ldots, T$: **do**
    For all examples x in $U \setminus S$:
        1. Compute its hypothetical label $\hat{y}(x) = h_{\theta_t}(x)$.
        2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{\mathrm{out}}} \ell_{\mathrm{CE}}(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$, where θ_{out} refers to parameters of the final (output) layer.
    Compute S_t, a random subset of $U \setminus S$, using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \setminus S\}$ and query for their labels.
    $S \leftarrow S \cup S_t$.
    Train a model θ_{t+1} on S by minimizing $\mathbb{E}_S[\ell_{\mathrm{CE}}(f(x;\theta), y)]$.
end for
return Final model θ_{T+1}.

Data	Class 1	Class 2	Class 3
X1	0.9	0.07	0.03
X2	0.87	0.03	0.1
X3	0.2	0.5	0.3
X4	0	0.01	0.99

Least Confidence

$$s_{LC} = \underset{x}{\operatorname{argmax}} \, (1 - P(\hat{y}|x))$$

$$\hat{y} = \underset{y}{\operatorname{argmax}} \, P(y|x)$$

Margin Sampling

$$s_{MS} = \underset{x}{\operatorname{argmin}} \, (P(\hat{y}_{max}|x) - P(\hat{y}_{max-1}|x))$$

Entropy

$$s_E = \underset{x}{\operatorname{argmax}} \left(-\sum_i P(\hat{y}_i|x) \log P(\hat{y}_i|x) \right)$$

L1 and L2

$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

Ridge (L1) regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity.

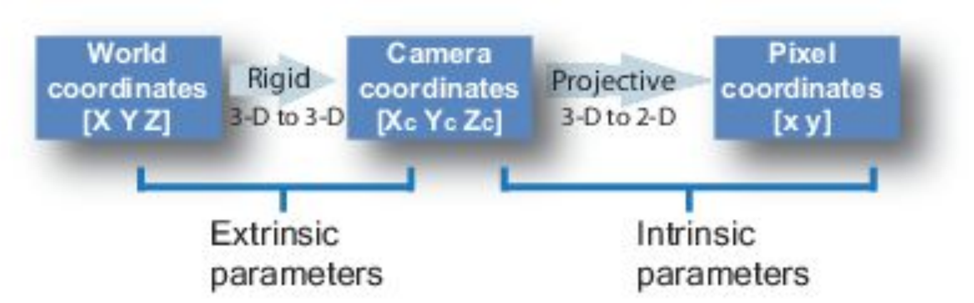
$$L2LossFunction = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

L2 regularization – feature selection

Calibration of camera

Intrinsic Parameters -

- Focal length (fx,fy) and optical center (cx,cy) for calculation of x,y in 3D scene - (x - cx) * Z / fx
- Depth correction factor for calculation of depth - depth/factor
- Distortion parameters - $x$distorted = $x(1 + k1*r2 + k2*r4 + k3*r6)$
- $y$distorted= $y(1 + k1*r2 + k2*r4 + k3*r6)$
- x, y — Undistorted pixel locations. x and y are in normalized image coordinates. Normalized image coordinates are calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels. Thus, x and y are dimensionless.
- $k1$, $k2$, and $k3$ — Radial distortion coefficients of the lens.
- $r2 = x2 + y2$



Extrinsic Parameters – Rotation and translation.

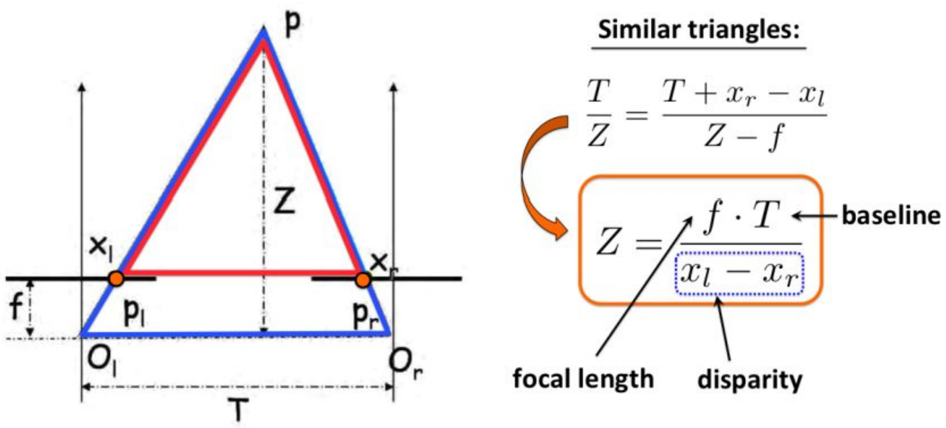
Epipolar Geometry -

Essential Matrix contains the information about translation and rotation, which describe the location of the second camera relative to the first in global coordinates.

The Fundamental Matrix contains the same information as the Essential Matrix in addition to the information about the intrinsics of both cameras so that we can relate the two cameras in pixel coordinates.

Epipolar geometry is used to find the depth of a point based on the camera feed of two cameras. Steps -

1. SIFT and RANSAC for finding the key points.
2. Construction of the fundamental matrix based on the key points.
3. Extraction of epilines and epipoles.
4. Based on this information depth of a point can be determined.
5. (Revise about SIFT and RANSAC)



Transformations:

1. **World-to-Camera:** 3D-3D projection. Rotation, Scaling, Translation
2. **Camera-to-Image:** 3D-2D projection. Loss of information. Depends on the camera model and its parameters (pinhole, f-theta, etc)
3. **Image-to-Pixel:** 2D-2D projection. Continuous to discrete. Quantization and origin shift.

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} f/\rho_u & 0 & c_x & 0 \\ 0 & f/\rho_v & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{Camera Intrinsic Matrix}} \underbrace{\begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{pmatrix}_{(4\times4)}}_{\text{Camera Extrinsic Matrix}} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}$$

Camera matrices — Image by Author

Depth from mono

- The goal in *monocular depth estimation* is to predict the depth value of each pixel or inferring depth information, given only a single RGB image as input.
- We will optimize 3 losses in our mode. 1. Structural similarity index(SSIM). 2. L1-loss, or Point-wise depth in our case. 3. Depth smoothness loss.

Major Loss Functions

- Cross entropy (classification & localization) - UNet
- Dice Coefficient (Localization)
- Shape Aware Loss (Localization)
- Structural Similarity Index (Depth estimation)

Loss Function

$$L = L_{cls} + L_{box} + L_{mask}$$

Extraction:

- Study code, Study indication, therapeutic Area, trial phase, healthy volunteer study, interventional study, pediatric study, Visits and schedule.
- Used python-docx, lxml, NER using BioBert

Ranking:

- Ranking by comparison of current study with past studies.
- Based on features like Study code, Study indication, therapeutic Area, trial phase, healthy volunteer study, interventional study, pediatric study.
- Study code – by giving major weightage to the start of study code and decreasing there on.
- Study indication and therapeutic area – Cosine similarity of biobert embeddings.
- Phase comparison using tanh
- Boolean comparison for healthy volunteer, interventional study and pediatric study.

STAR example

- **S**ituation: Set the scene and give the necessary details of your example.
  Task: Describe what your responsibility was in that situation.
  Action: Explain exactly what steps you took to address it.
  Result: Share what outcomes your actions achieved.
- Situation Example: In my previous role, As my team lead contracted with COVID, I volunteered to take the responsibility of managing the Backend team.
- Task Example: The responsibility was to smoothly carry out day to day backend team activities.
- Action Example: In order to that, I had to be very careful about how I managed all of my time. So, I used to set aside some time each day for each of the tasks. During that time I collaborated with business analyst for requirement analysis, stakeholders for user acceptance testing and frontend team for integration of backend APIs.
- Result Example: The project was able to meet all the deadlines. In recognition to my contribution I received ACE award and promoted to Senior Analyst in less than 2 years.

Intro

Hello, I am Ratan. I am currently pursuing a Master's of Computer Science with Data Science Specialization at North Carolina State University.

My coursework and projects at NCSU are majorly related to Data Science, Deep Learning, Natural Language processing and Computer vision.

I am currently doing a part time as a data scientist for the crop and soil science department in assisting for their precision sustainable agriculture project. Creating data pipelines for image data collected from agricultural fields.

In terms of professional experience,

I have started of my professional career with research internship at Hewlett-Packard and worked on research of data structures and Machine Learning for firmware of printers. This created an interest for me in the field of programming and data science.

After this I took up a role with Accenture as an analyst. At Accenture, I was part of a team which developed an application for generation of clinical trial documents for a pharmaceutical client. In this role, I had to collaborate with various stakeholders like business analyst, architects, end users for building backend modules for the application. I have designed and developed various modules from scratch using python, NLP and Azure cloud services.

During my time at Accenture, I felt the urge of learning more into technologies like AI and data science, hence I have decided to purse Master’s of computer science with data science specialization at NCSU.

Random Forest vs GBTrees

- Random Forest Vs Gradient Boosting -
  The main difference between random forests and gradient boosting lies in how the decision trees are created and aggregated. Unlike random forests, the decision trees in gradient boosting are built additively; in other words, each decision tree is built one after another.
- Gradient Boosting -
  - Initialize the base model by taking argmin of the loss function.
  - Calculate the residual for the base model.
  - Build decision trees for the features on residuals.
  - Iterate the steps.

Test by sending the inputs into the base model and constructed decision trees and combine the trees with the specified learning rate.(Should have a differentiable loss function.)

Gradient Boosting -
Eg - XGBoost.

- Each new tree is built to improve on the deficiencies of the previous trees and this concept is called *boosting*.
- The *gradient* part of gradient boosting comes from minimizing the gradient of the loss function as the algorithm builds each tree.
- Boosting focuses step by step on difficult examples that give a nice strategy to deal with unbalanced datasets by strengthening the impact of the positive class.
- Training generally takes longer because of the fact that trees are built sequentially.
- GBMs are harder to tune than RF. There are typically three parameters: number of trees, depth of trees and learning rate, and each tree built is generally shallow.

Random Forest -

- RFs train each tree independently, using a random sample of the data. This randomness helps to make the model more robust than a single decision tree, and less likely to overfit on the training data
- The real-world data is noisy and contains many missing values, some of the attributes are categorical, or semi-continuous.
- We need high predictive accuracy for a high-dimensional problem with highly correlated features.
- RF is much easier to tune than GBM. There are typically two parameters in RF: number of trees and number of features to be selected at each node.
- The main limitation of the Random Forests algorithm is that a large number of trees may make the algorithm slow for real-time prediction.

Linear Regression –

Independent variables should not be co-linear.

Doesn’t handle outliers.

Supports only linear data.

Performs well for less data aswell.

Fast.

Logistic Regression

Easy, fast and explainable.

For linear data.

Co-linearity not desirable.

Doesn’t handle categorical values well.

Performs well for less data as well.

KNN-

Slow for large data.

Careful selection of k.

Scaling required.

Is non-parametric.

Works well with high data.

Decision Trees-

No preprocessing

Handles collinearity

Explainable, fast

Overfitting if pruning is not performed

Prone to outliers.

Not suitable for continuous variables.

Hyperparameters – criterion, max depth, min samples to split, min samples at leaf.

SVM-

Hinge loss

Kernel trick.

Handles outliers

Tough hyperparameter tuning.

Long training.

Not suitable for multi-class.

Doesn’t give probability.

Convex function – goes to global minima.

Naïve bayes-

Expects independent features

Less training data

SVM – C penalty

Gamma similarity

3D CV →

RGB-D ⇄ Point Cloud

through Camera Calibration

Point Cloud
↓ scaling to 32x32x32
Voxel Grid
↓ CNN (Voxnet*)
(classification)

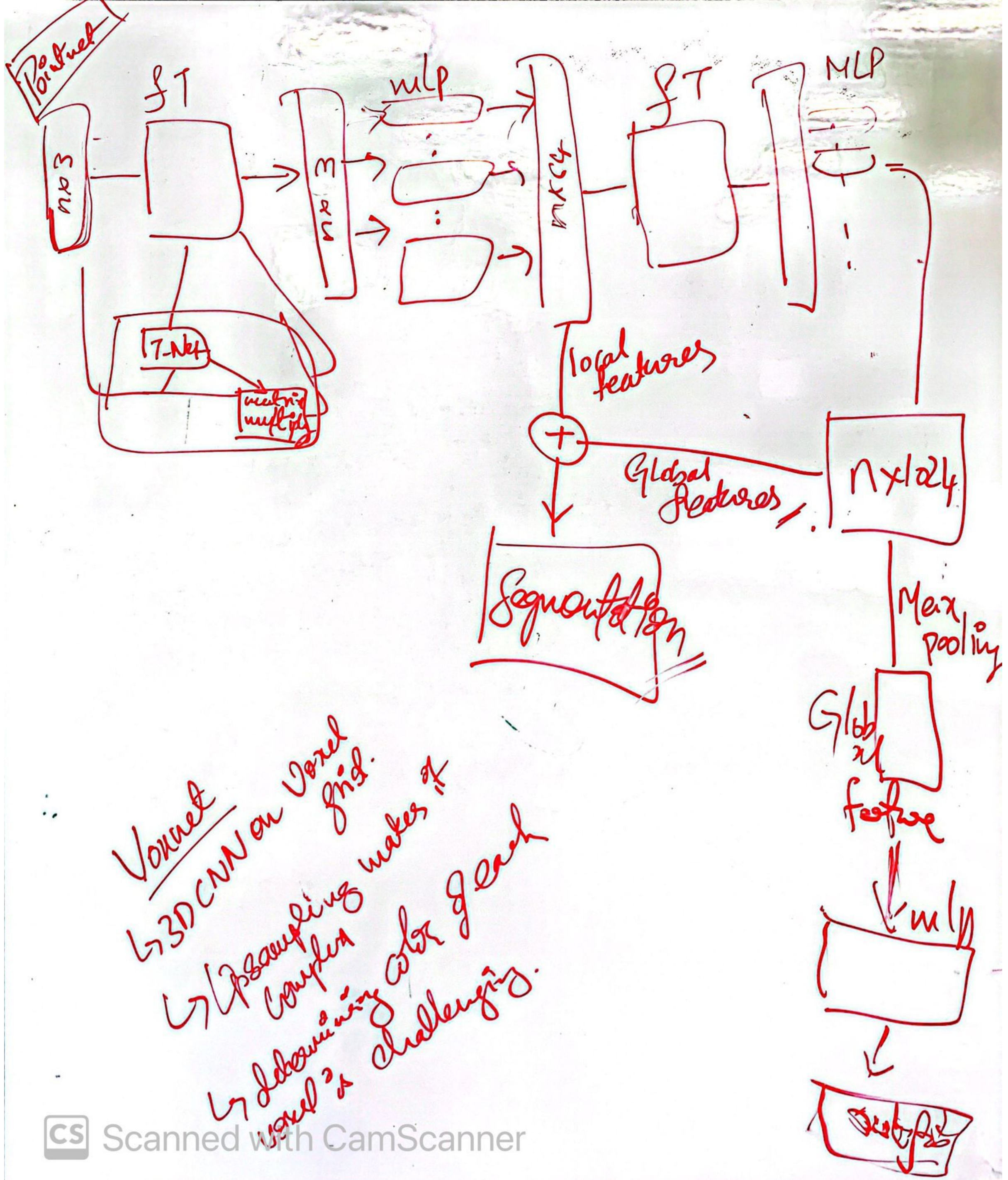
3D Scatter plot for visualizing Point Clouds.

Segmentation - Categorical Cross entropy loss

Major Loss functions
- Cross Entropy
- Dice loss (IOU)
- Shape Aware loss

Point Cloud
↓
Point Net
↓
Classification.

- Permutation invariant. — Solved by using max-pooling (Global feature space)
- Training by randomly sampling the points.
- Transformation invariant. — Solved by Spacial Transformation Network.



Voxnet
↳ 3D CNN on Voxel grid.
↳ Upsampling makes it complex
↳ determining color of each voxel is challenging.

GANs

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

Inception Score
Test generated samples on pre-trained inception model.

$$\begin{aligned}\mathrm{IS}(x) &= \exp(\mathbb{E}_{x\sim p_g}[D_{\mathrm{KL}}[p(y|x) \parallel p(y)]]) \\ &= \exp(\mathbb{E}_{x\sim p_g, y\sim p(y|x)}[\log p(y|x) - \log p(y)]) \\ &= \exp(H(y) - H(y|x))\end{aligned}$$

Frechet Inception distance – compare the feature vector's Mean and covariance of GAN with Inception

---

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

---

for number of training iterations **do**
  for k steps **do**
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Object Tracking

- Motion Model - It predicts the potential position of objects in the future frames, hence, reducing the search space. Eg – KLT, optical flow.
- Visual Appearance Model – Tracking based on the appearance of the object
- ROLO – Recurrent YOLO
- MDNet, GOTURN

Motion Detection

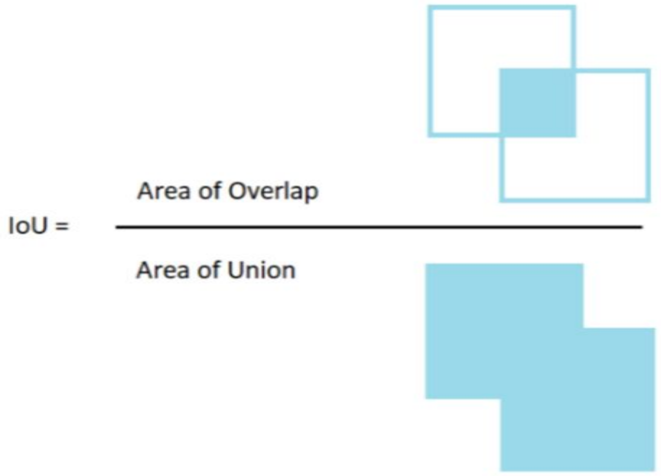
- Background subtraction (subtract current image with a static background)
- Feature Matching
- Optical flow – illumination is critical
- Template Matching – for stabilizing videos

Optical Flow Equation

$$I_x u + I_y v + I_t = 0$$

Jaccard Score

$$Jacc(y, \hat{y}) = \frac{\sum_{c}^{C} y_c \hat{y}_c}{\sum_{c}^{C} y_c + \hat{y}_c - y_c \hat{y}_c}$$



Visualization of Jaccard score

Loss functions and formula

Categorical Cross Entropy

$$CE = -\sum_{i=1}^{i=N} y\_true_i \cdot log(y\_pred_i)$$

$$CE = -\sum_{i=1}^{i=N} y_i \cdot log(\widehat{y_i})$$

$$\Longrightarrow CE = -[y_1 \cdot log(\widehat{y_1}) + y_2 \cdot log(\widehat{y_2}) + y_3 \cdot log(\widehat{y_3})]$$

Loss Function

The multi-task loss function of Mask R-CNN combines the loss of classification, localization and segmentation mask: $\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}}$, where $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{box}}$ are same as in Faster R-CNN.

The mask branch generates a mask of dimension m x m for each RoI and each class; K classes in total. Thus, the total output is of size $K \cdot m^2$. Because the model is trying to learn a mask for each class, there is no competition among classes for generating masks.

$\mathcal{L}_{\text{mask}}$ is defined as the average binary cross-entropy loss, only including k-th mask if the region is associated with the ground truth class k.

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} [y_{ij} \log \hat{y}^k_{ij} + (1 - y_{ij}) \log(1 - \hat{y}^k_{ij})]$$

where y_{ij} is the label of a cell (i, j) in the true mask for the region of size m x m; $\hat{y}^k_{ij}$ is the predicted value of the same cell in the mask learned for the ground-truth class k.

Loss Function

Faster R-CNN is optimized for a multi-task loss function, similar to fast R-CNN.

| **Symbol** | **Explanation** | | p_i | Predicted probability of anchor i being an object. | | p^*_i | Ground truth label (binary) of whether anchor i is an object. | | t_i | Predicted four parameterized coordinates. | | t^*_i | Ground truth coordinates. | | N_{cls} | Normalization term, set to be mini-batch size (~256) in the paper. | | N_{box} | Normalization term, set to the number of anchor locations (~2400) in the paper. | | λ | A balancing parameter, set to be ~10 in the paper (so that both $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{box}}$ terms are roughly equally weighted). | {:.info}

The multi-task loss function combines the losses of classification and bounding box regression:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$

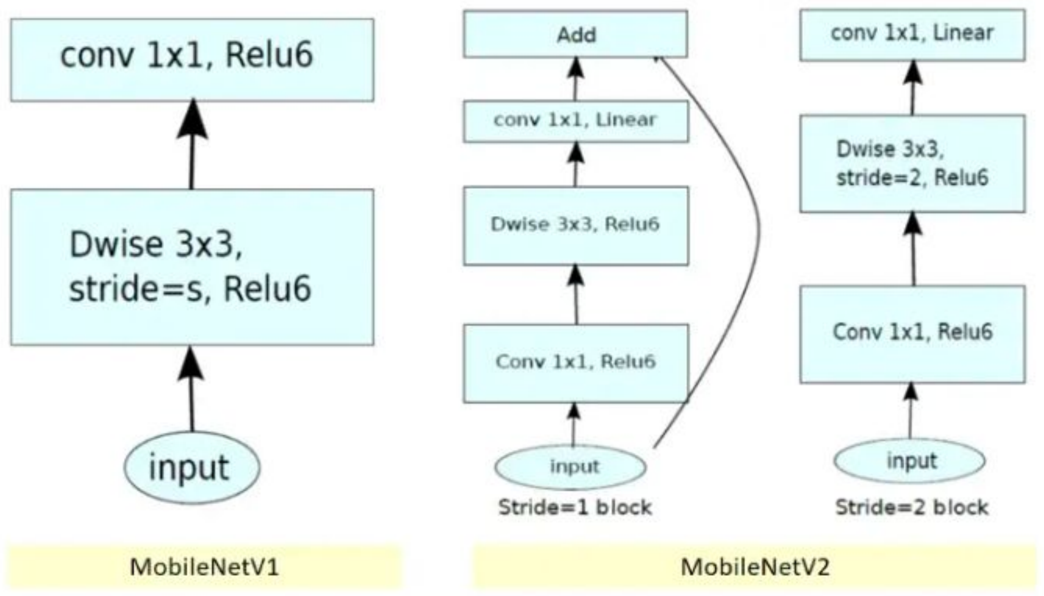
$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}(p_i, p^*_i) + \frac{\lambda}{N_{\text{box}}} \sum_i p^*_i \cdot L_1^{\text{smooth}}(t_i - t^*_i)$$

where $\mathcal{L}_{\text{cls}}$ is the log loss function over two classes, as we can easily translate a multi-class classification into a binary classification by predicting a sample being a target object versus not. L_1^{smooth} is the smooth L1 loss.

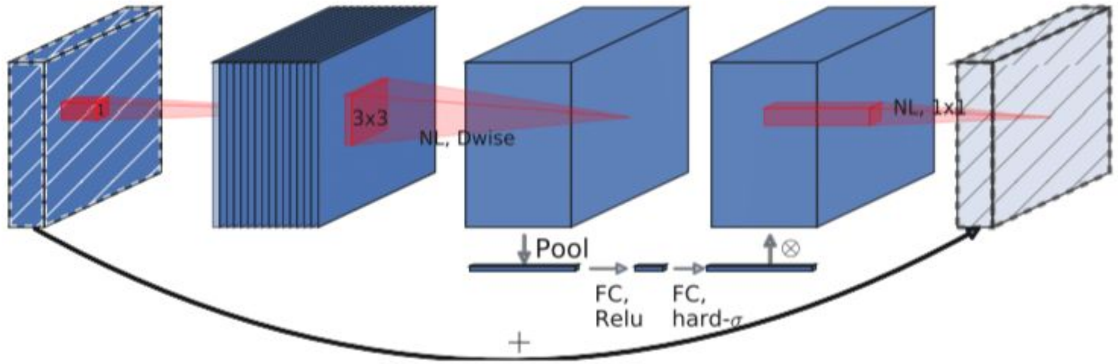
$$\mathcal{L}_{\text{cls}}(p_i, p^*_i) = -p^*_i \log p_i - (1 - p^*_i) \log(1 - p_i)$$

YOLO – classification loss + localization loss + confidence loss

MobilenetV2



Mobilenetv3



- Product workflow:
- SWOT and Porter's 5 Analysis
- Product Market Positioning - alpha release, Voice of the customer - Surveys, Focus Groups, Online Forums
- Marketing Plan - Gap Analysis (Opportunities), Conjoint Analysis (Important Features), Total Addressable Market
- Feasibility Assessment - Technical Feasibility, Market Feasibility, Cost Feasibility
- Forecasting - ATAR (Awareness, Trial, Availability, Reuse)

Strengths 1. What is our competitive advantage? 2. What resources do we have? 3. What products are performing well?	**Weaknesses** 1. Where can we improve? 2. What products are underperforming? 3. Where are we lacking resources?
Threats 1. What new regulations threaten operations? 2. What do our competitors do well? 3. What consumer trends threaten business?	**Opportunities** 1. What technology can we use to improve operations? 2. Can we expand our core operations? 3. What new market segments can we explore?