Target Recognition using Embeddings from Radar Cross-Section

BTP Phase 2

BTP Group Code:- HSS_3

- ***Narendra Pal (160108028)***
- ***Kevin Jose (160108047)***

Under the guidance of ***Dr. Hanumant Singh Shekhawat***

EEE Department, IIT Guwahati Assam-781039

June 2020

Outline

Introduction

Information Sources from Targets

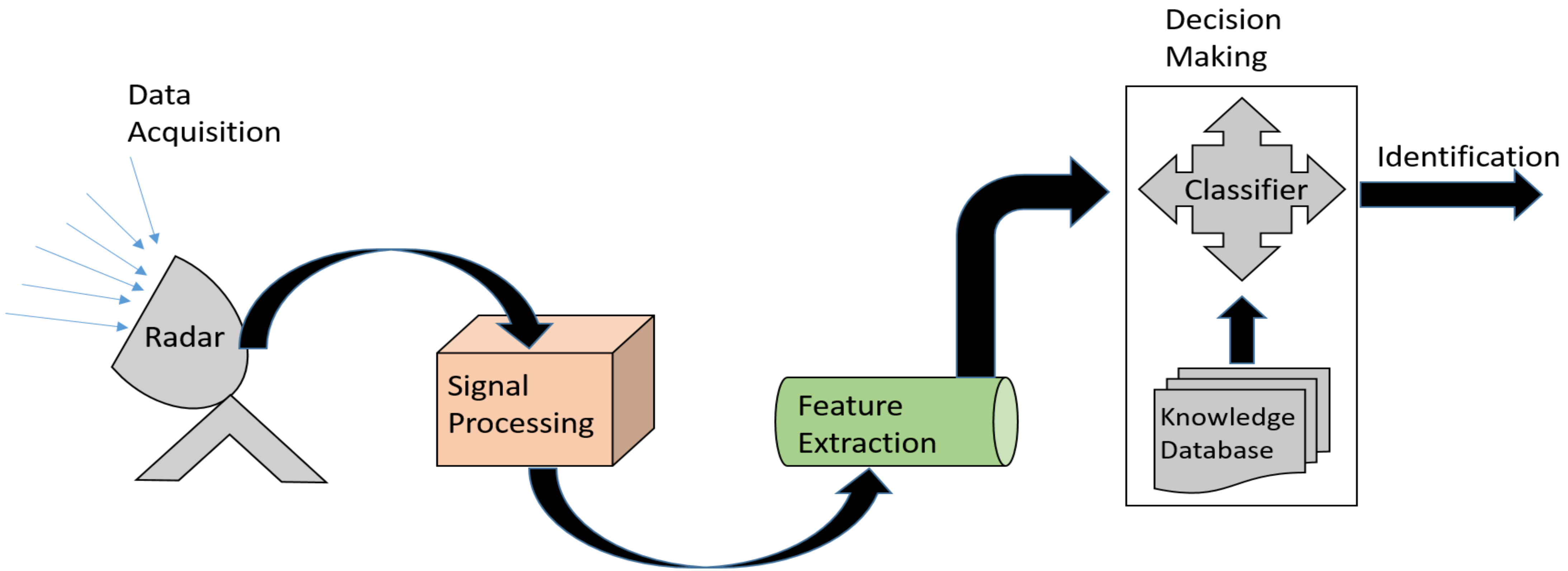
1. Navigational data

- Target velocity
- Direction
- Trajectory

2. Target geometry

- Radar cross section(RCS)
- Rotor blade
- Fixed wings

Introduction



<u>Problem Statement</u>

“Design a complete model which can identify an object/target with its RCS data with high accuracy irrespective of its orientation and the direction of motion.”

Existing Technique for Target Identification

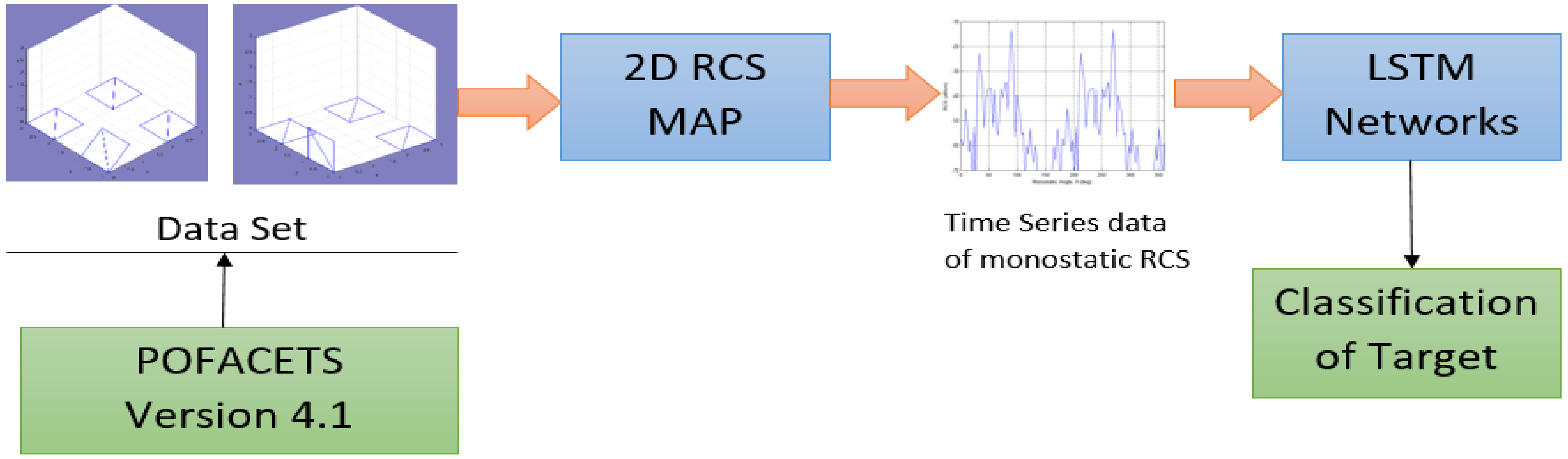
- Machine Learning and Synthetic Aperture Radar (SAR) based

It uses the concept of a synthetic aperture to create high-resolution images of the target object and predicting the class of a target object using SAR images has become easier with machine learning techniques.

- Multi-Static radar Based

Multi-static radars utilize multiple mono-static or bi-static radars. The area of coverage and spatial diversity of a multi-static radar is far more than that of a simple monostatic radar. Thus, multi-static radars prove to be very useful in target detection, parameter estimation (with non coherent processing), and target location (with coherent processing).

Model Implemented in BTP phase 1

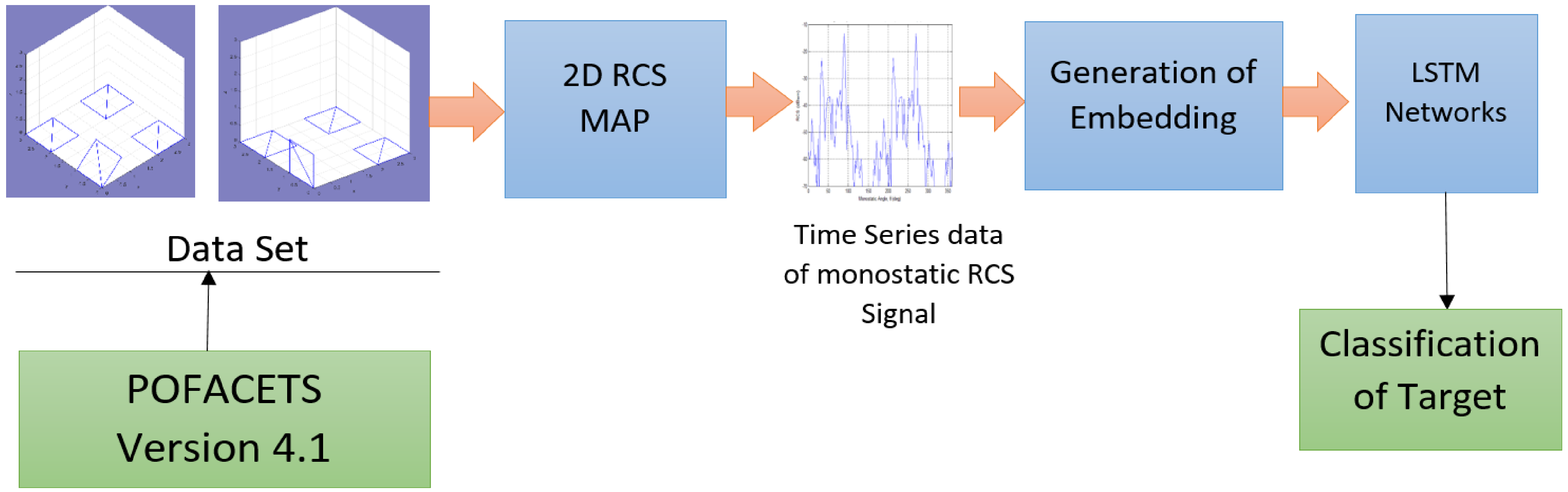


In this model we got the favorable result with an accuracy of 98.6%, but result seems to be over-fitted.

Drawbacks of these models:

All the existing models and the model we implemented in BTP phase 1 fail when targets are in a different orientation or a different direction of motion with respect to the mono-static radar system, which is the case in real life.

Proposed Approach



Literature Review

Radar Cross-Section (RCS)

- RCS of a target is an effective area that receives the transmitted power emitted by radar, and it reflects back power isotropically to the Radar system.
- It gives information on how detectable an object is by the radar system.
- Larger the value of RCS indicates that the target is easily detectable, lesser the value is less likely to be detected.
- Some of the factors on which RCS value depends:
  - The material of which the target is made.
  - Size of target with respect to the wavelength of the radar signal.
  - The absolute size of the target.
  - The orientation of target with respect radar system.
  - Incident and Reflected angle of the radar signal.

Mathematical Representation of RCS:

Mathematical equation to calculate the value of RCS:

$$S_r = \frac{S_t * G}{4\pi r^2} * (rcs) * \frac{1}{4\pi r^2} * A_{eff}$$

S_r is Power received by the radar system

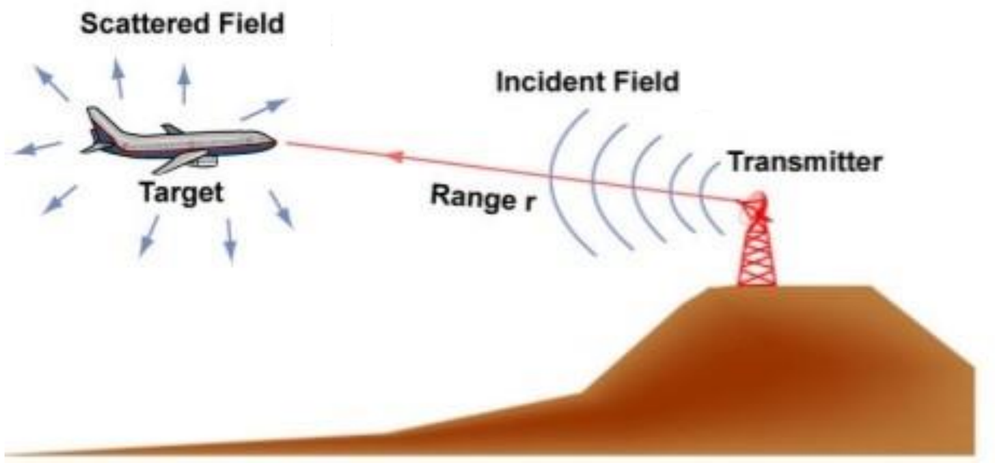
S_t is Power emitted by the radar

G is Gain value of the radar antenna

r is distance of target from the radar system

rcs is RCS value of the radar target

A_{eff} is effective area of radar at which signal is received



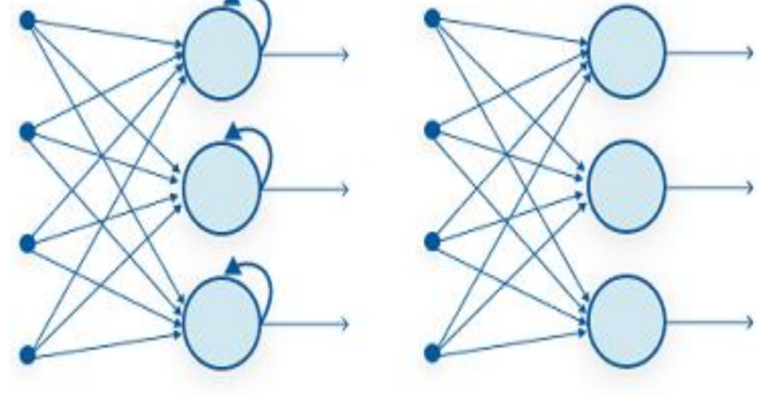
However, In this project we are not using this formula to get the value of RCS, rather by using POFACETS tool on MATLAB to generate this value.

Narrowband Radar

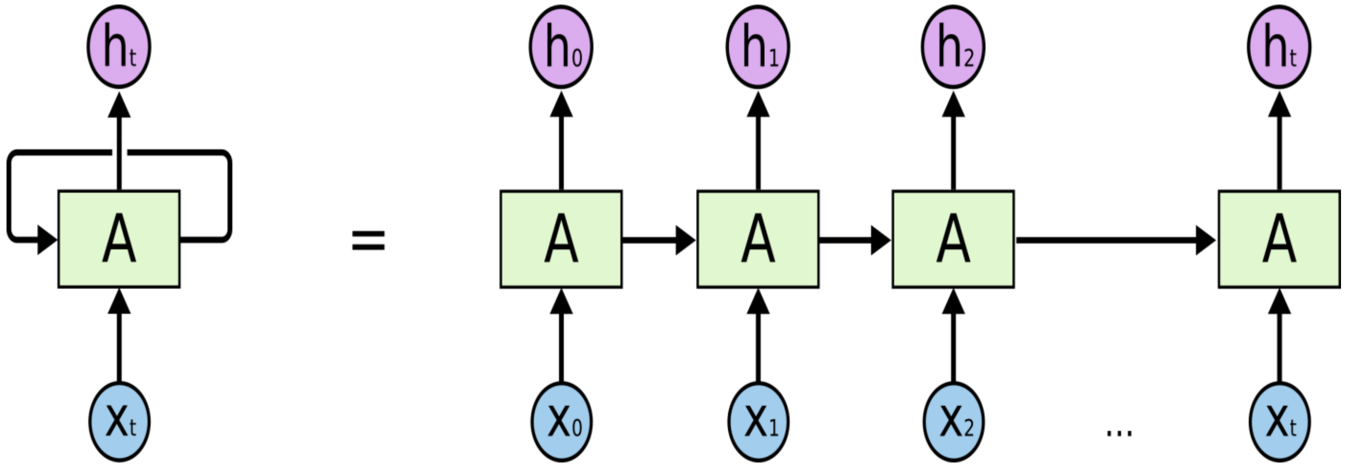
- This radar systems are emitting sinusoidal or quasi-sinusoidal signals towards the target.
- These sinusoidal and quasi sinusoidal signals can easily be generated by the RLC oscillator circuit.
- In this project, We have simulated the narrowband monostatic radar and 3D objects using the POFACETS MATLAB tool.
- The time series of monostatic RCS generated using this tool does not contain noise as compared to a time series generated by a real narrowband radar.
- In order to apply the classification method proposed in this project for a real radar, we add Gaussian noise to change our training and testing dataset a little bit to maintain signal to noise ratio.

Recurrent Neural Network (RNN)

- We can classify neural networks into two types: General Feed-Forward Networks(FFN) and Recurrent Neural Networks(RNN) with feedback connections.



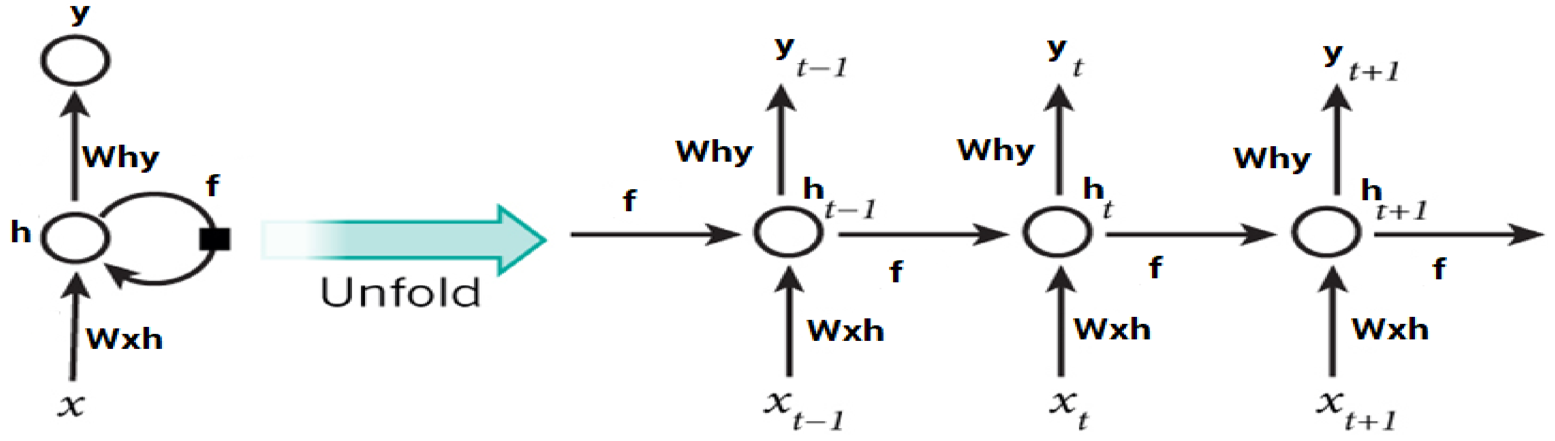
Recurrent Neural Network and Feed-Forward Neural Network



Unrolling a single node of RNN into a series of Feed-Forward Neural Network

- In feedforward neural networks, the data passes through each point only once in an iteration. As there are no feedback connections, there is no possibility of memory.
- RNN is FFN with a time twist: neurons are fed information not just from the previous layer but also from themselves from the previous pass.

RNN(Contd.)



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

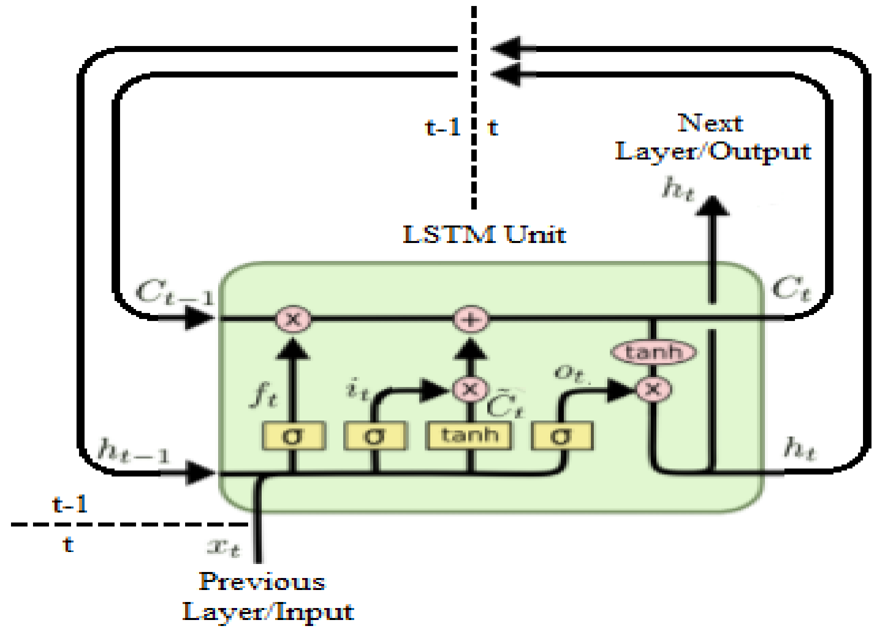
- A general Feed-Forward Neural Network is trained using forward propagation, error estimation using loss functions and, back propagation. But in RNNs, we use Back Propagation Through Time (BPTT). It is just back-prop through all the time steps for each node.

Problem with RNN :

- In practice RNNs are limited to looking back only a few steps.
- It can not efficiently address the long term dependencies i.e. suffers from vanishing gradient while BPTT.

Long Short Term Memory Network (LSTM)

- Long Short Term Memory is some special kind of RNNs which can learn long term dependencies, by picking it up, saving it and injecting it back to the network whenever and wherever necessary.



$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, X_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

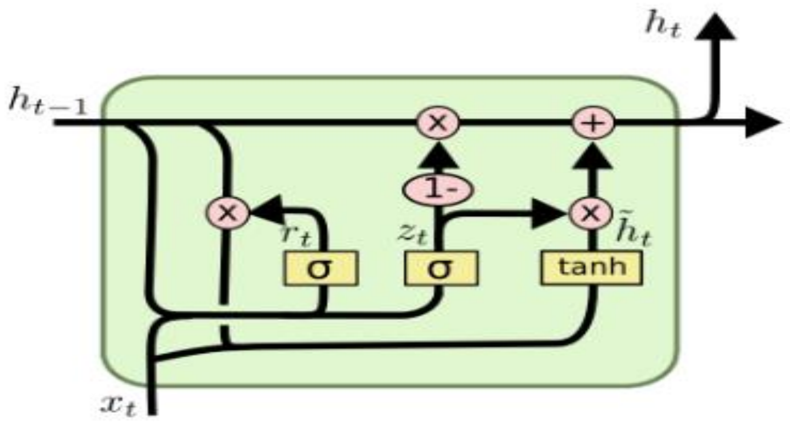
σ is sigmoid function

tanh is hyperbolic tan function

- What makes it different from conventional RNN architecture ?
  - Due to presence of cell gate, forget gate , input gate, output gate it makes the LSTM a special RNN which is capable of learning long term dependencies.

Gated Recurrent Units(GRUs)

- GRUs are a modified version of LSTMs to address the problem of vanishing gradient.
- It use two gates, namely, update gate and reset gate to address this problem.
- GRU has, in fact, lesser parameters than LSTMs, and it does not have an output gate.



$$Z_t = \sigma(W_z \cdot [h_{t-1}, X_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, X_t])$$

$$\bar{h}_t = \tanh(W \cdot [r_t * h_{t-1}, X_t])$$

$$h_t = (1 - Z_t) * h_{t-1} + Z_t * \bar{h}_t)$$

σ is sigmoid function

tanh is hyperbolic tan function

- They can be trained to keep information from a long time ago without losing it and also to remove unwanted formation from cell states.

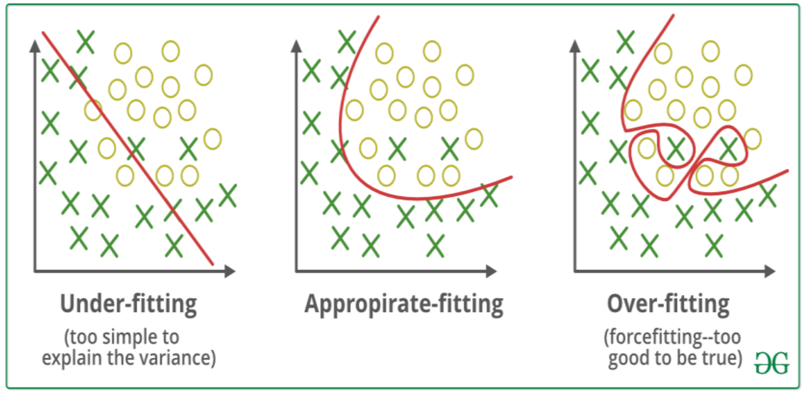
Regularization

- When we train any model with a lot of data, then the overfitting issue came in the model, which means it starts learning with noise as well, and it does not perform well on new data. To overcome this problem, we use Regularization (by adding penalty).

❑ L2 regularization

It adds a penalty in the form of a squared magnitude of coefficient.

$$Loss = Error(y - \bar{y}) + \lambda \sum_{i=1}^{N} W_i^2$$


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

❑ L1 Regularization

It adds a penalty in the form of the absolute value of coefficients.

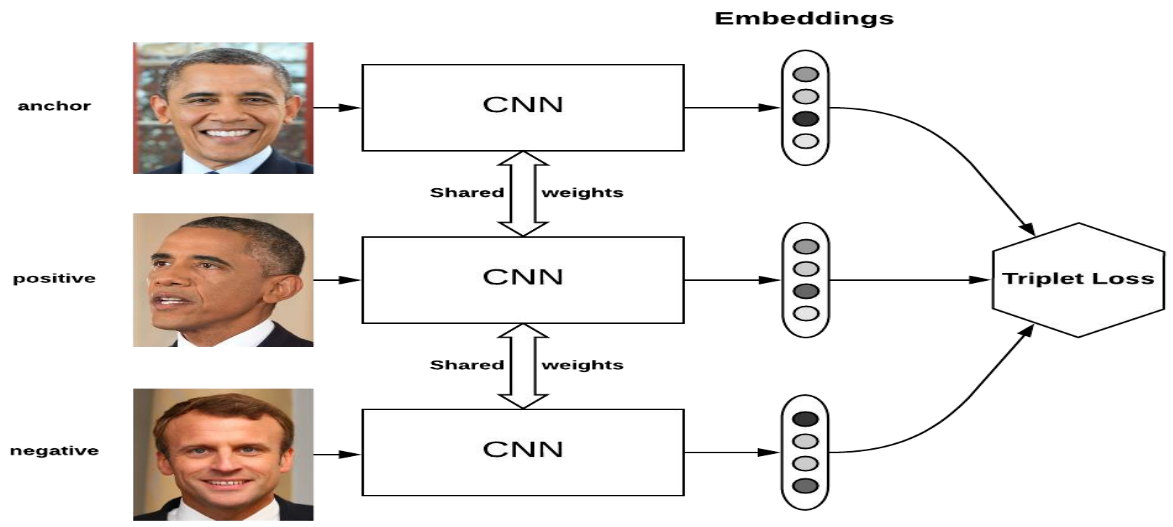
$$Loss = Error(y - \bar{y}) + \lambda \sum_{i=1}^{N} |W_i|$$

❑ Max Norm Regularization

It executed by scaling the magnitude of weight vector of every neuron not more than a particular value.

FaceNet

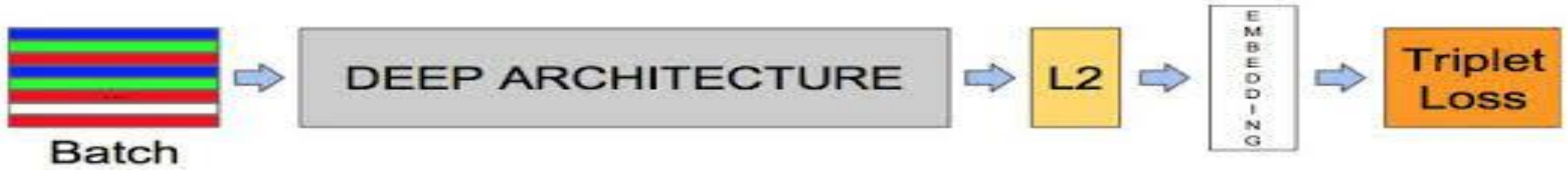
- A team of researchers from Google proposed a facial recognition system named FaceNet in 2015 which attains a good result on face recognition.
- It uses a deep learning architecture such as Zeiler & Fergus(ZF-Net) and Inception model to create a good quality face mapping from images.
- It maps the images to Euclidean space such that distance in the Euclidean space correspond to face similarity.
- Loss function called as Triplet Loss is used for training the model.



- Here in our project, we have used RCS value visualized as an image instead of images in mapping and train a similar model to generate embedding.

FaceNet(Contd.)

- **Embedding:**
  - An embedding can be generally understood as a mapping of a discrete categorical variable to a vector of a definite dimension of continuous numbers.
  - Embeddings are less dimensional and learned continuous vector representations of discrete variables in the Neural network context.
  - These generated embeddings are very useful because they can reduce the dimensionality of categorical variables and can represent meaningfully, categories in the transformed space.
- **Basic Outline of the model:**



  - It uses deep Convolution Neural Network, and network is trained in such a way that squared distance between two RCS correspond to its similarity.
  - The aim of this triplet loss function is to make the squared distance independent of the condition and orientation of the same object small between the embeddings

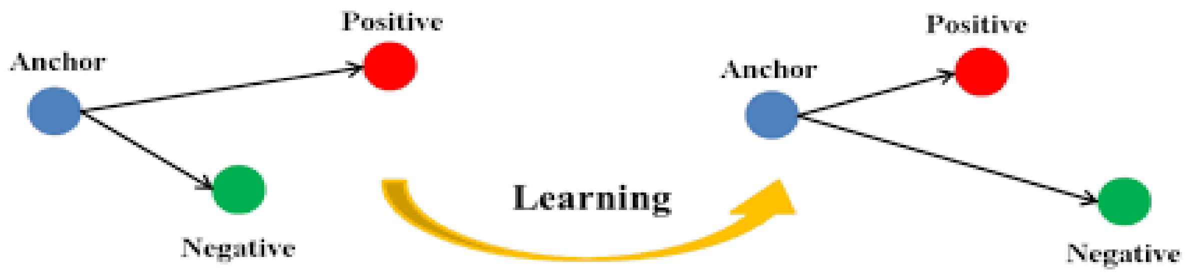
FaceNet(contd.)

- Triplet Loss Function:
  - To measure the losses, Triplet Loss function uses three set of RCS that is positive - of the same object, anchor - close to positive , negative - RCS of another object.

$$Loss = \sum_{i=1}^{N} [\ ||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]$$

x_i = representation of an input RCS data

$f(x_i)$ = representing an embedding of the RCS data.



Below expression is the constraint used to make it certain that x_i^a (anchor) is closer to x_i^p (positive) compared to x_i^n that of another object(negative).

$$||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2$$

FaceNet(contd.)

- **Triplet Selection:**

- Selecting the right pairs of RCS is highly important as there are a lot of pairs of RCS value that will fulfill the above constraint condition and therefore our model will not learn much from them and will converge slowly because of all this.
- To make the faster learning of model, it finds the triplet which violate the above constraints i.e. for given anchor it selects the hard positive value of $||f(x_i^a) - f(x_i^p)||_2^2$, and hard negative value of

  $||f(x_i^a) - f(x_i^n)||_2^2$.

- Generating triplets based on an entire training set is computationally very expensive. So, to overcome from this problem below points are the two modes for triplet generation.
  1. Based on earlier checkpoints, creating triplets on each step, and calculating minima and maxima on a subset of data.
  2. By computing the hard positive and hard negative for a mini-batch such as 1500-2000 samples.

Methodology and Results

Defining Target Class

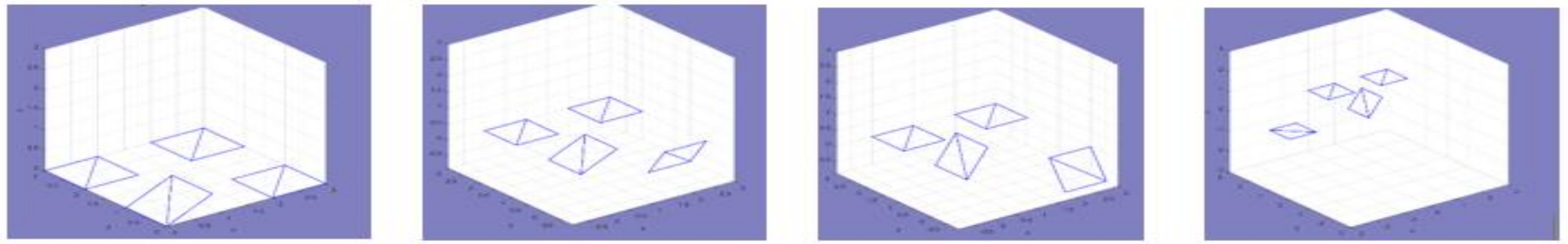
- In this project, we have considered four different types of target classes. Each target class consists of four plates with material properties given below:

$$Perfect\ \ Electrical\ \ conductor$$

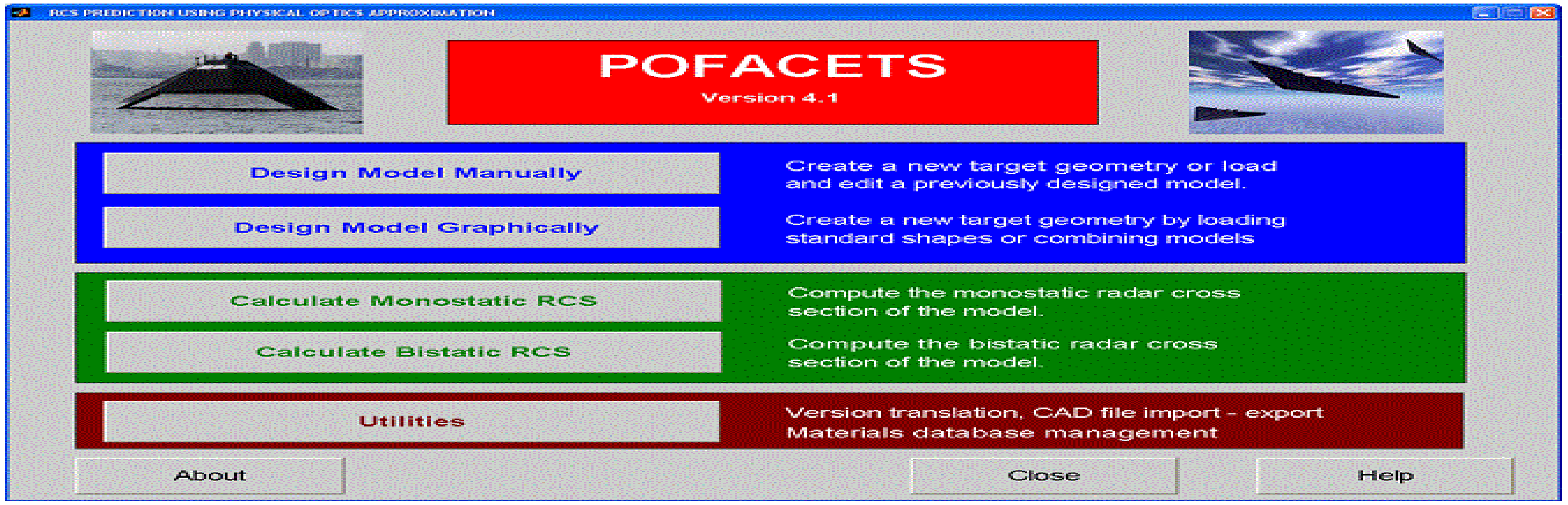
$$Loss\ \ tangent = 0.0045$$

$$Permeability = 1$$

$$Thickness\ \ of\ \ plate = 5mm$$


Orientation of Plates at different angle

- To get the monostatic RCS value at each configuration, we have used POFACETS, a MATLAB tool.


RCS PREDICTION USING PHYSICAL OPTICS APPROXIMATION

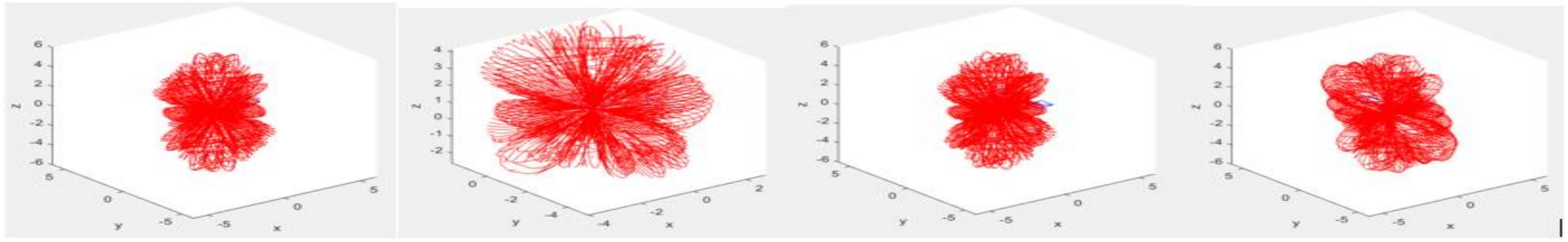
POFACETS
Version 4.1

Design Model Manually	Create a new target geometry or load and edit a previously designed model.
Design Model Graphically	Create a new target geometry by loading standard shapes or combining models
Calculate Monostatic RCS	Compute the monostatic radar cross section of the model.
Calculate Bistatic RCS	Compute the bistatic radar cross section of the model.
Utilities	Version translation, CAD file import - export Materials database management

About Close Help

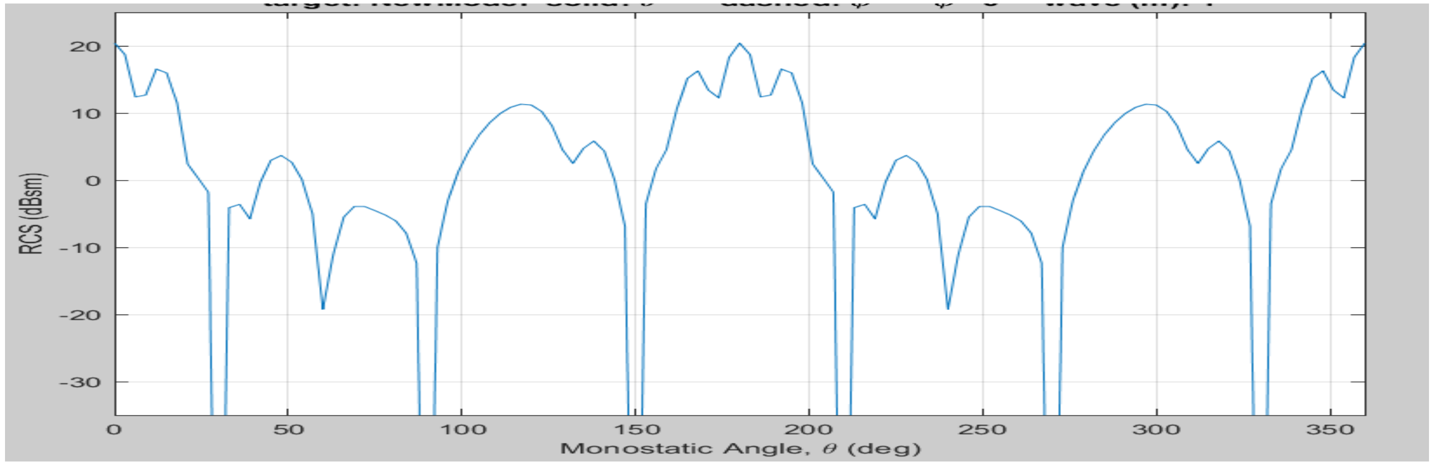
Generating Data Set

- After getting the RCS value, we created the 3D map of the target.



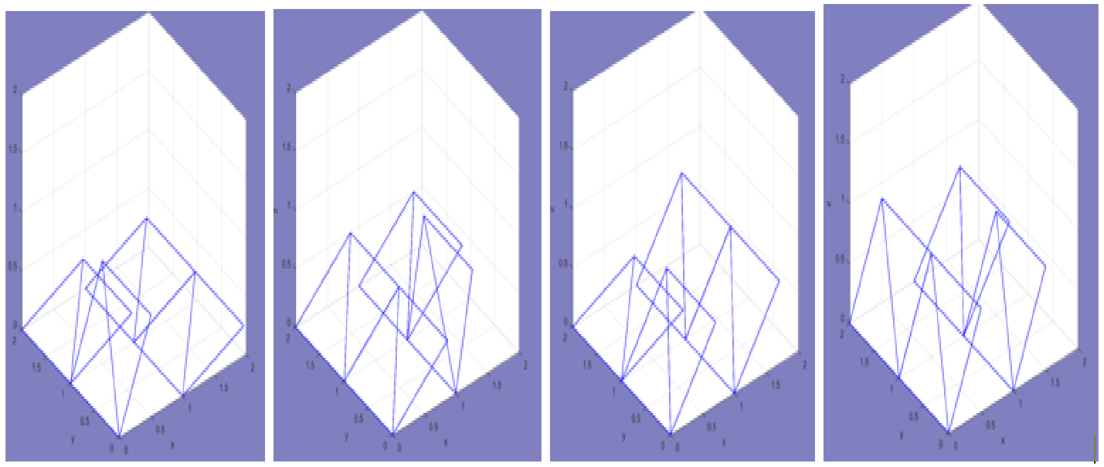
3D RCS map for different objects

- 2D RCS maps is created with POFACETS , In this 2D map, mapping is done within the angular range with a sampling interval of Ø is 3 degree and Ɵ is 1 degree. then we generate the time series value of monostatic RCS values. And we fed this signal to LSTM based model(done in BTP phase1 for classification which attains an accuracy of 98.6%.

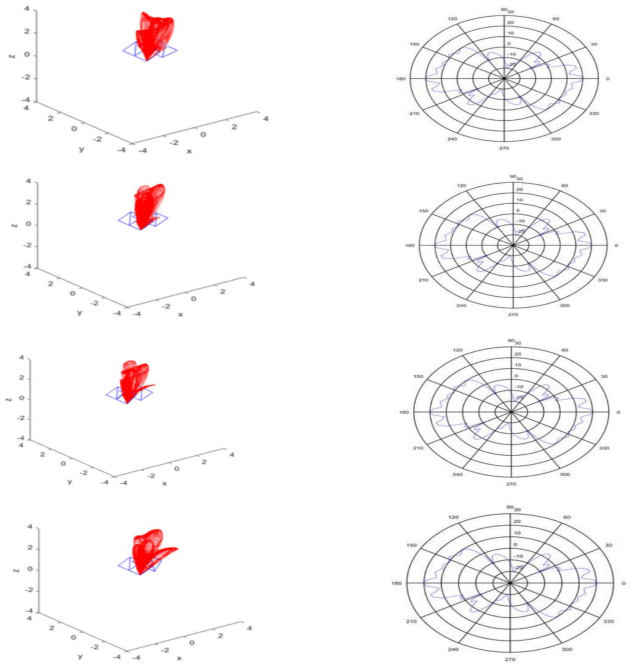


Generating Data Set(Contd.)

- However, this model fails when aircraft is at some angle i.e. orientation of plates are not parallel to the surface. To tackle this problem we change the orientation of whole system(in this phase of our project) at some angle as shown below and use it to train a model to generate embedding.



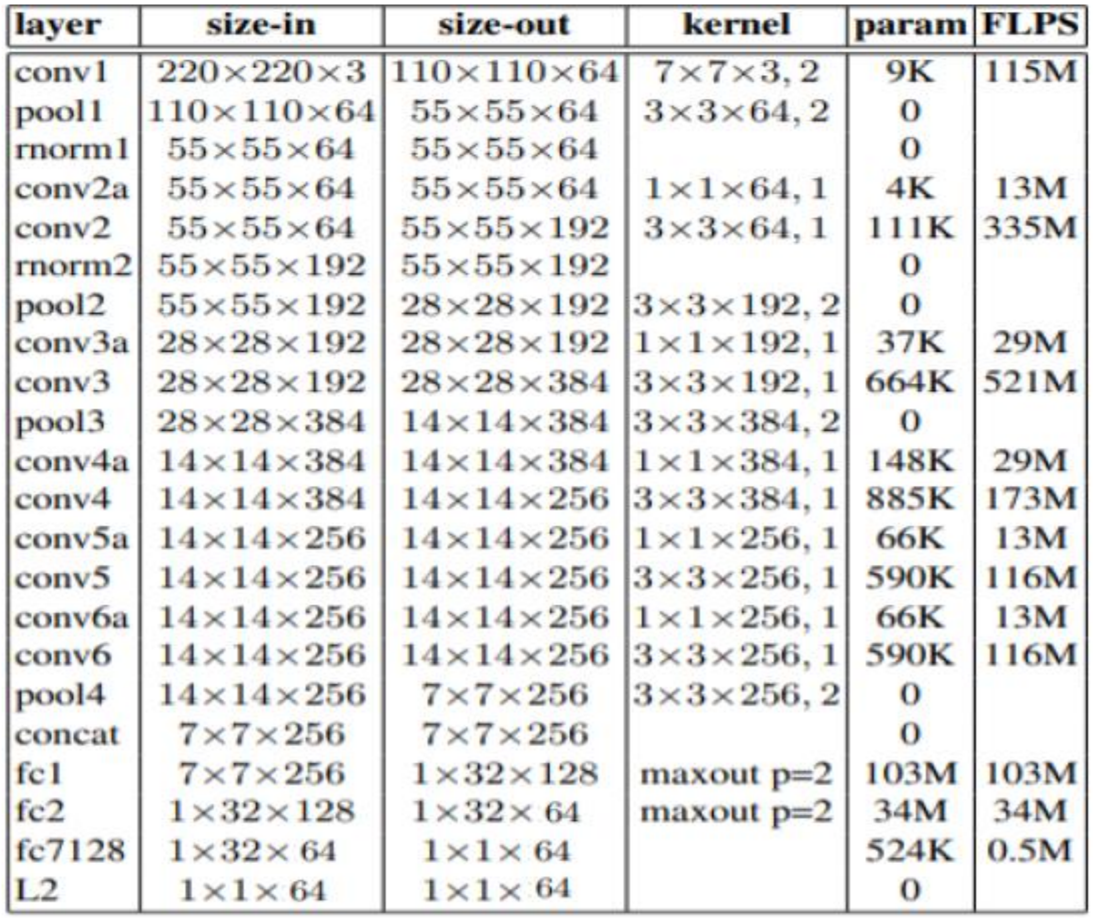
Different orientation of plates in 4 different target class



3D RCS plot and its corresponding polar graph of different orientations

Generating Embeddings

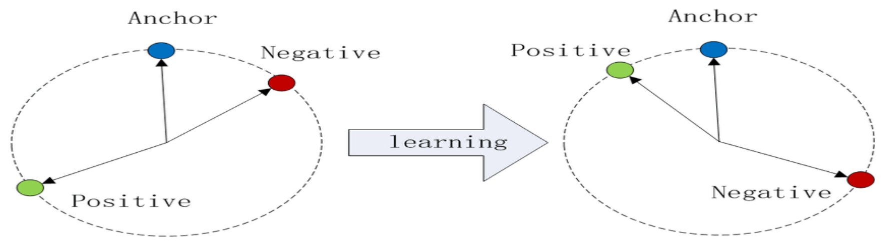
- After the dataset is generated and defined, we need to define a model for generating embedding. Our proposed model is as shown in the table below.

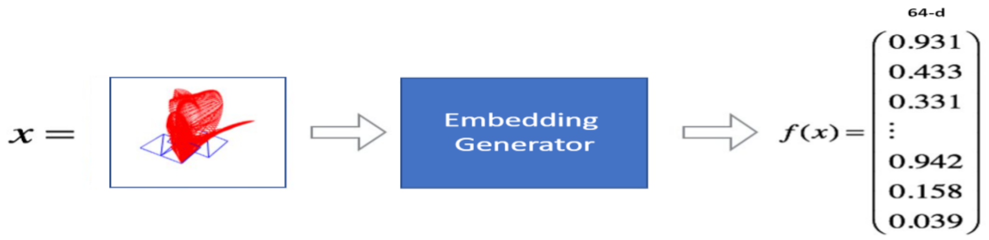


layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32× 64	maxout p=2	34M	34M
fc7128	1×32× 64	1×1× 64		524K	0.5M
L2	1×1× 64	1×1× 64		0	

Generating Embeddings(contd.)

- For now we have assumed that a 64 x 1 embedding vector will satisfy our criteria. For training, we manually picked anchor, positive and negative data according to the conditions and fed into the model to generate embeddings. This model is trained to minimize the ”distance” between like objects and maximize the ”distance” between unlike objects as shown below.





- For training, we can use a setup known as Siamese Network, which is a another network with shared weights with our current network. So we have 3 Siamese networks here, where we feed in anchor ,positive and negative respectively in each one of them as input and each of them generates its own embedding.
- These embedding are used to calculate triplet loss. As we can see, triplet loss function is differentiable everywhere. So we can use back propagation to change the weights of the shared network. Our objective is to minimize the Triplet loss function.
- once we have the embedding generator ready, we feed in input from the test set and accuracy is observed. If our model is successful in generating an accurate embedding, we can feed this directly into the stacked LSTM and if needed further training can also be done. For now, we generated embeddings for each training class and stored it separately.
- For testing, we generated embeddings of the test RCS and manually compared that with the embeddings already in our database. If the ”distance” between the embeddings is less than a threshold, we classified it as like objects.

Results

As of now, we used the pre-trained model similar to FaceNet and used transfer learning to train an additional 2 layers to generate embeddings for our dataset. After making a database of embeddings, we generated embeddings from test set and with a threshold of 4, we got a classification accuracy of 79.6%.

Conclusion and Future Work

❑ Conclusion

- We found that classification based on RCS values is possible and can be achieved with high accuracy.
- To make our classification invariant to the object’s condition and orientation, instead of feeding in directly the RCS, we generated embeddings.
- The classification accuracy is comparatively low as compared to the existing models, but it can be observed that the performance is better to other contemporary models when same objects with different orientations are subject to classification.

❑ Future work

- we suggest retraining the complete embedding generator with a slightly different model to obtain higher accuracy since FaceNet is inherently designed for facial recognition purpose.
- Since we used the same model for feeding in the RCS values directly as used in Phase 1, this part also has a scope for improvement.
- We would also like to suggest employing Generative Adversarial Networks (GANs) to generate inverse mapping from RCS values to the objects.

Thank you !