Beginning 2020, I started to explore different ML applications, specifically NLP and Computer Vision. I want to explain my journey to learn Computer Vision and one of the projects that I have worked on.
I started with some courses on coursera to learn about the fundamentals of deep learning, Neural Networks and Deep Learning and then Convolutional Neural Networks. I learned alot about the fundamentals of deep learning and computer vision. As a practice for these courses, I started working on Fashion MNIST dataset (https://github.com/zalandoresearch/fashion-mnist) to apply CNN algorithms to classify different kinds of clothings and fashion items. To expand my skills, I started studying another deep learning series of courses, Deep learning Tensorflow developer. These series of courses focus on the using tensorflow and keras to develop enhanced computer vision models.
After taking these courses I started to work on a project that was a collaboration with a startup company that creates radar sensors for vehicles. As a volunteer in this collaboration, I wanted to apply the techniques I had learned to achieve the best results. I'll explain the process in the following paragraphs.

Radar Target Classification Using the Deep Convolutional Networks

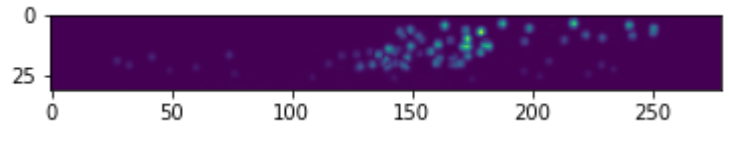
With the recent development of autonomous driving, there's been a lot of work on detecting moving targets in the roads with radar and camera. Target detection will help autonomous vehicles companies to have better road planning. One of earliest studies for autonomous driving on classification shows above 90% accuracy of other vehicles, pedestrians, bicycles and other moving objects (1)
With the development of deep learning, these numbers on accuracy and precision have been increased but there's a lot of ongoing research on improving these results.

In this project I've used FMCW Radar data to classify road moving targets including cars, bicycles and pedestrians. In the future, I'd like to develop the data set with other moving objects, so the algorithm can detect more moving targets by training different classes of objects.

Data

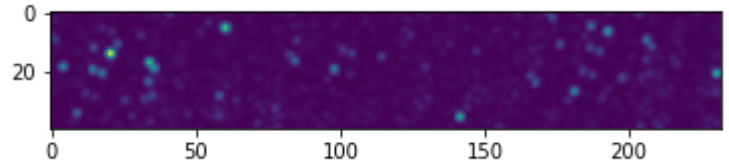
For this project I have used 900 training samples from 3 different categories in the data set. This data has been chosen by permutation among 39000 samples per class.
(I have repeated this process 5 times to build 5 different training set and apply the model on them to be sure about the accuracy of the model)
Total number of validation samples was 11700 samples per class which I have 150 samples from each class and for the test set I have chosen a set of 22800 samples from different classes to verify the accuracy and precision metrics of the model.

Samples of the data:

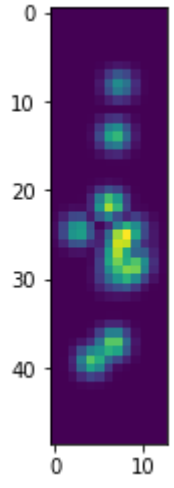


sample of bike radar image

sample of car radar image



sample of car radar image



sample of pedestrian radar image

Tools and Technologies

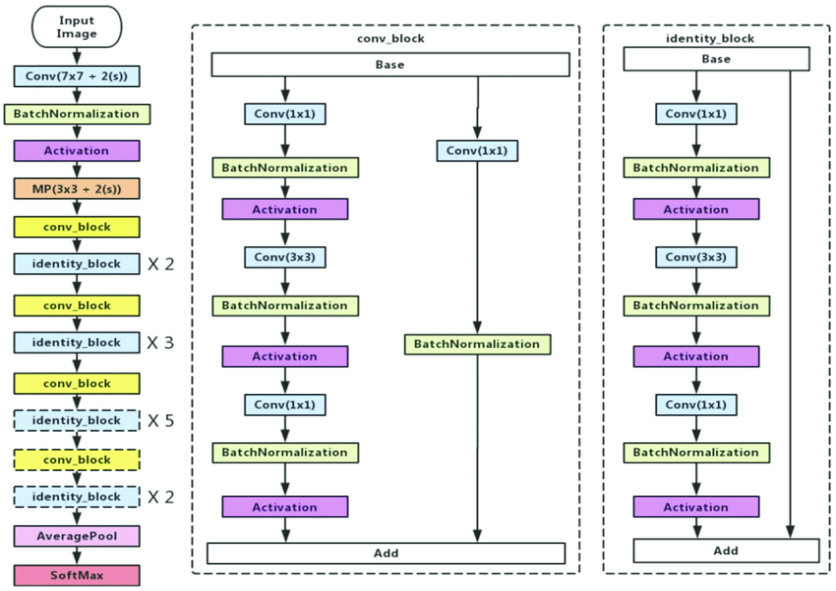
For this project I have used google colaboratory, tensorflow (https://www.tensorflow.org/) and keras which is the high-level API of TensorFlow 2.0. It is an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. (https://keras.io)

Model

For developing the model that best fits this dataset, I have trained different convolutional neural network models to compare the training and validation results. From combination of conv layers with pooling and dense layers to more complicated ones. Some of the architectures that I've trained on this data set are: Inception (2), ResNet (3), VGG (4) and Xception (5).

From these models, I have shared the model with the ResNet-50 architecture in this document and I'll explain the results that I got by training ResNet-50 on the radar data set.

ResNet (Deep Residual Learning for Image Recognition) is one the models for image classification with weights trained on imagenet. And it is used very frequently in many computer vision tasks.
The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters.
(https://arxiv.org/abs/1512.03385)



ResNet-50 Architecture

Image Source
(https://www.researchgate.net/figure/Left-ResNet50-architecture-Blocks-with-dotted-line-represents-modules-that-might-be_fig3_331364877)

Training the model

With ResNet-50 architecture as the backbone of the model, I have added the dense layer for the classification with kernel regularizer and initializer in order to get the best result possible.

```
# channel_last
# input shape = (128, 128, 3)
from tensorflow.keras import initializers
base_modelin=tf.keras.applications.DenseNet169(include_top=False, weights='imagenet', input_shape=(128,128,3), classes=3
xin=base_modelin.output
xin=tf.keras.layers.GlobalAveragePooling2D(data_format=None)(xin)
#x=tf.keras.layers.Dropout(0.5)(xin)
xout=tf.keras.layers.Dense(3, kernel_regularizer=tf.keras.regularizers.l1_l2(l1=1e-5, l2=1e-5),
                           activation = tf.nn.softmax,
                           kernel_initializer = tf.keras.initializers.RandomNormal(mean=0., stddev=0.0001))(xin)
model=tf.keras.models.Model(inputs=base_modelin.input, outputs=xout)
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy', tf.keras.metrics.AUC(), tf.keras.metrics.Precision()])
```

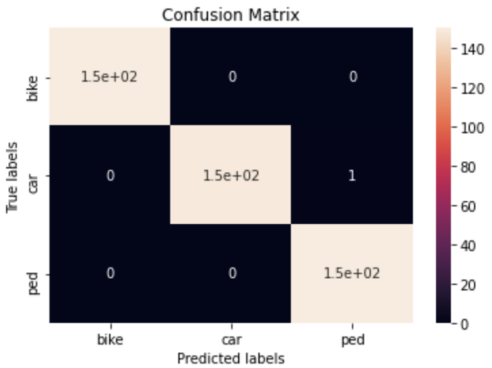
And also in order to avoid overfitting the model, I have used keras callback method which monitors the validation loss function and after 5 rounds of training, if the validation loss doesn't change, will do the early stopping on the training set.

Model Evaluation Metrics

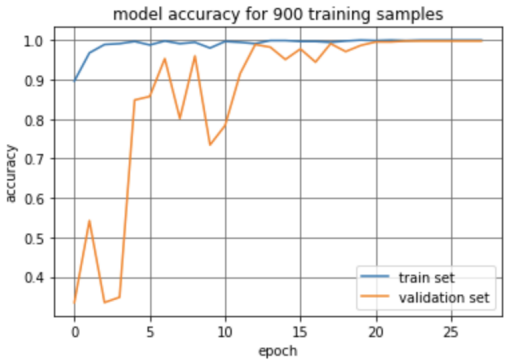
For evaluation metrics I have used accuray, precision and Area Under Curve(AUC) on both validation and test sets.

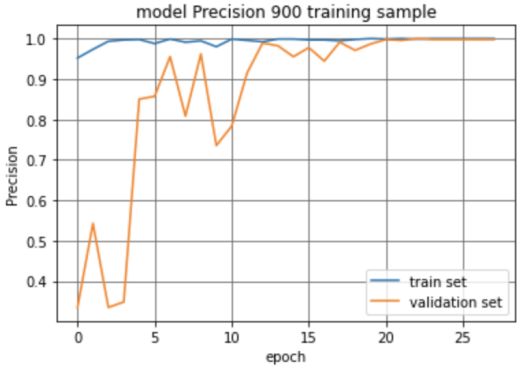
confusion matrix on validation set:

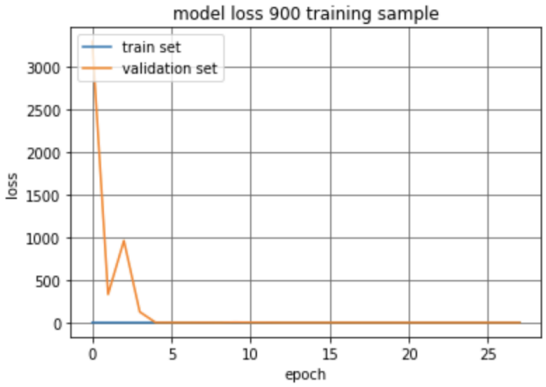
```
Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       150
           1       1.00      0.99      1.00       150
           2       0.99      1.00      1.00       150

    accuracy                           1.00       450
   macro avg       1.00      1.00      1.00       450
weighted avg       1.00      1.00      1.00       450
```

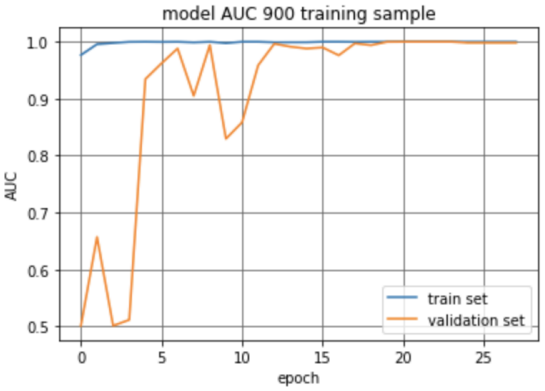


Accordingly, Accuracy, Precision, Loss and AUC for training and validation set:







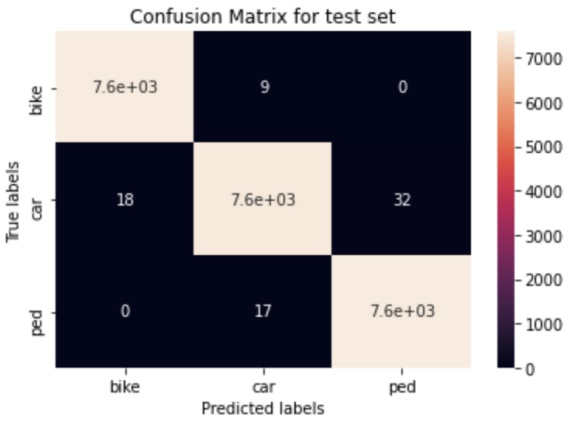


Evaluation of the model on the test set:

For evaluation of the model on the test set, I have tested the model on the 22800 Samples of the data. The result shows the great score of 0.99 percent precision in each class of data including the car, pedestrian and bikes.

Confusion Matrix for test set:

```
Classification report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7600
           1       1.00      0.99      0.99      7600
           2       1.00      1.00      1.00      7600

    accuracy                           1.00     22800
   macro avg       1.00      1.00      1.00     22800
weighted avg       1.00      1.00      1.00     22800
```



Future Work

For the future work, My plan is to work on data segmentation and boundary box for radar images and combine it with the classification part.
Image segmentation tasks involve detecting individual objects within the image. These objects can be buildings, roads, cars, or pedestrians. After detecting different objects in a radar image, it's the next step to label each object with an associated class as I've described in my project.

References

1- S. Heuel, and H. Rohling. "Two-stage pedestrian classification in au- tomotive radar systems," *12th International Radar Symposium (IRS),* 2011, pp. 477-484.
2- https://arxiv.org/abs/1512.00567v3
3- https://arxiv.org/abs/1512.03385
4- https://arxiv.org/abs/1409.1556
5- https://arxiv.org/abs/1610.02357