RF-based Direction Finding of UAVs Using Deep Neural Network

Anabel Reyes Carballeira (anabel.carballeira@mtel.inatel.br)

Eligário Milton da Costa Semedo (eligario@mtel.inatel.br)

National Institute of Telecommunications – Inatel

Master in Telecommunications

1st Semester, 2020

Introdução

- Violação de privacidade
- Ações criminais: roubo ou invasão domiciliar
- Tráfico de drogas dentro e fora das prisões
- Colisão e queda de drones nas pessoas.

- Em janeiro de 2015, um drone caiu na Casa Branca

- Em março de 2016, um jato da Lufthansa chegou a 200 pés de colisão com um drone perto do Aeroporto Internacional de Los Angeles

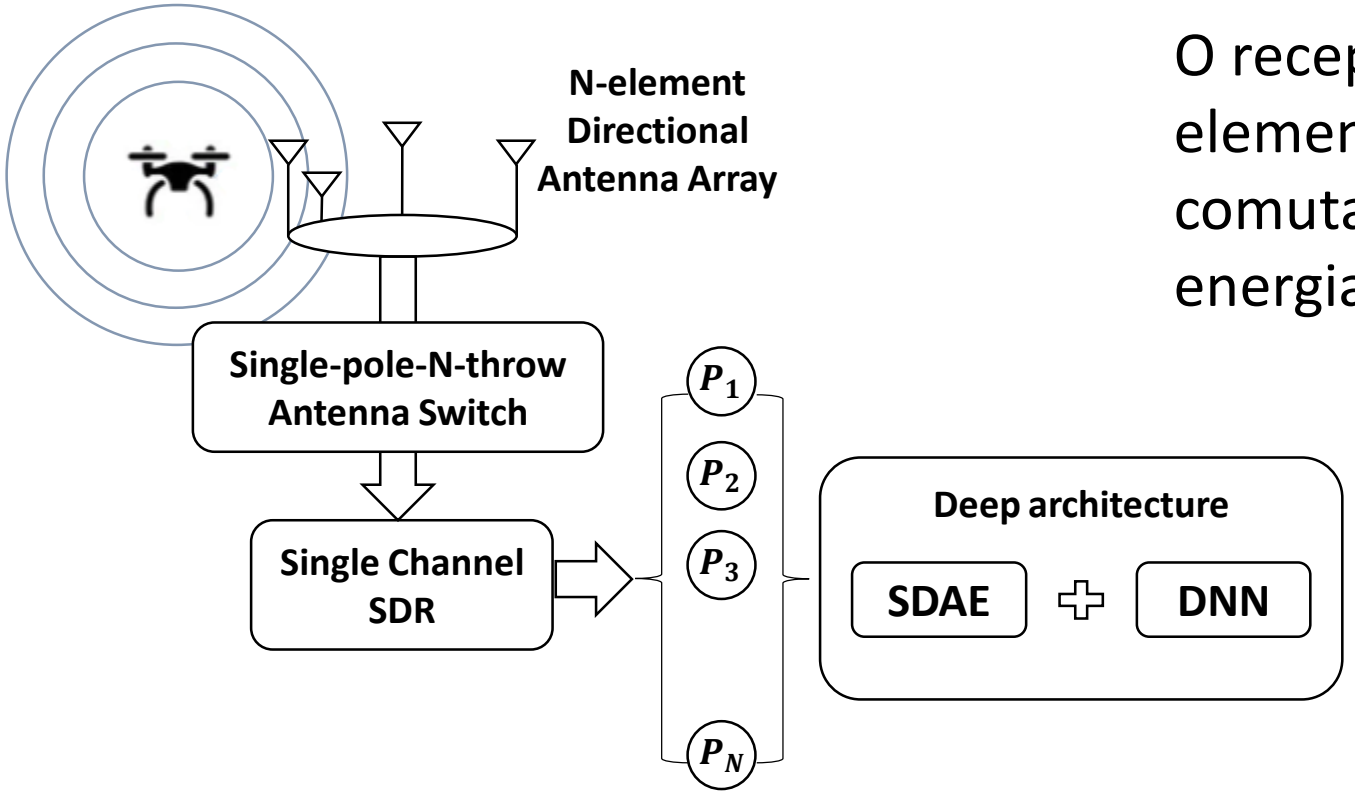
Introdução

- **As técnicas de detecção de direção (DF) baseadas em Radiofrequência foram bem estudadas e as técnicas clássicas de alta resolução,** como **MUSIC** e **ESPRIT.**
- Uso de comunicação **WiFi** como o **OFDM** (que são as técnicas mais usadas para comunicação por drones) que apresentam alguns desafios bem como a **complexidade do hardware e o consumo de energia**.

Objectivo

Propomos um método **prático e de baixo nível de complexidade** para **encontrar a direção** do drone (DF)

Sistema proposto para localização de direção de UAVs baseados em RF



O receptor ativa seqüencialmente um elemento de antena de cada vez usando o comutador RF-SPNT e mede o valor de energia recebido correspondente.

Goal

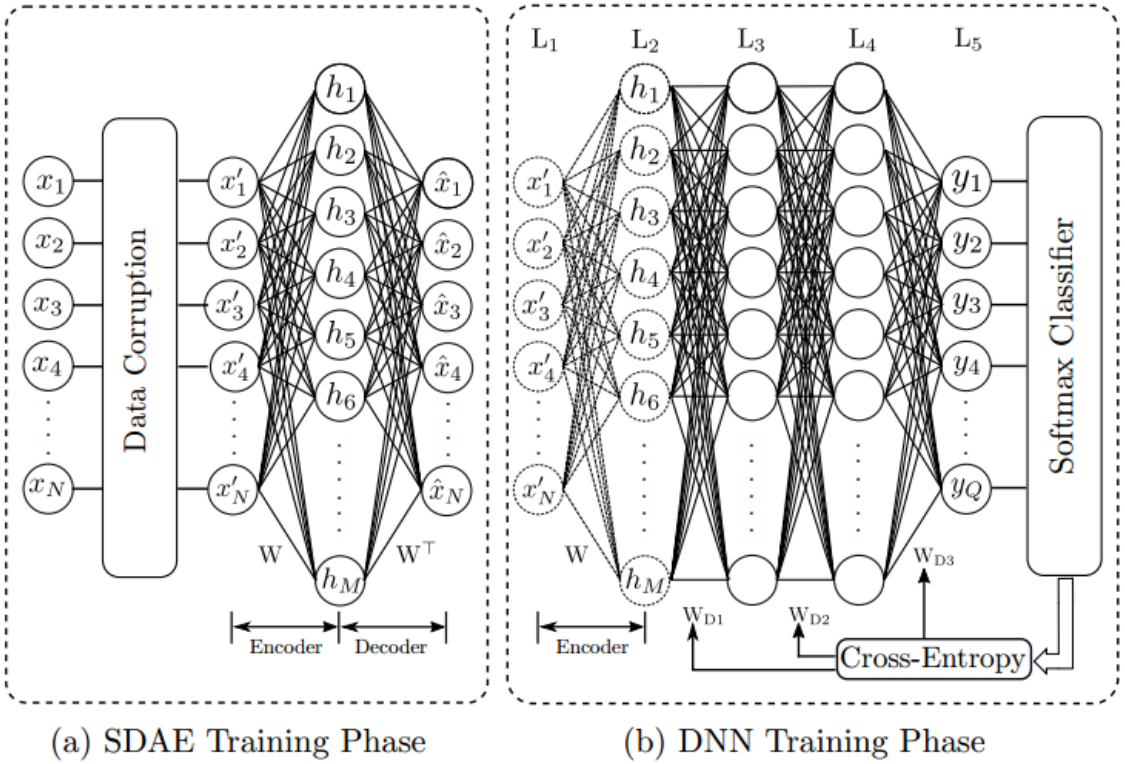
Obter um relacionamento subjacente (ou um padrão) entre a potência média do sinal recebido no conjunto da n-ésima antena $\left(\{\boldsymbol{P_n}\}_{\boldsymbol{n=1}}^{\boldsymbol{N}}\right)$ e o ângulo do azimute $\boldsymbol{\theta}$.

Algoritmo SDAE

1. Durante a fase de treinamento do SDAE, os valores de potência recebidos pré-processados **são atribuídos às unidades de entrada.**
2. Em seguida, os valores das unidades **da camada oculta e da camada de saída** são calculados como:
   $h = f(Wf_c(x) + b_e)$ e $x\_ouput = f(W^T h + b_d)$
3. Os parâmetros de SDAE (W, be e bd) **são otimizados** de forma que o erro de reconstrução seja minimizado, enquanto está sujeito a uma restrição de esparsidade.

$$L(\mathbf{W}, \boldsymbol{b_e}, \boldsymbol{b_d}) = \sum_{i=1}^{T} (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i})^2 + \beta \sum_{m=1}^{M} \mathrm{KL}(\rho || \rho_m),$$

$$\rho_m = \frac{1}{T} \sum_{i=1}^{T} h_m(\boldsymbol{x_i})$$



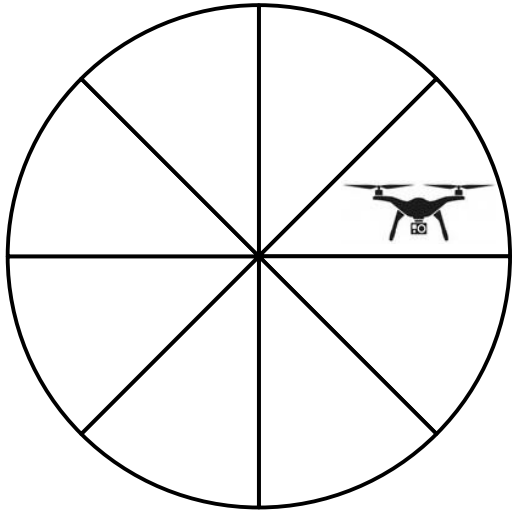
(a) SDAE Training Phase (b) DNN Training Phase

Algoritmo DNN

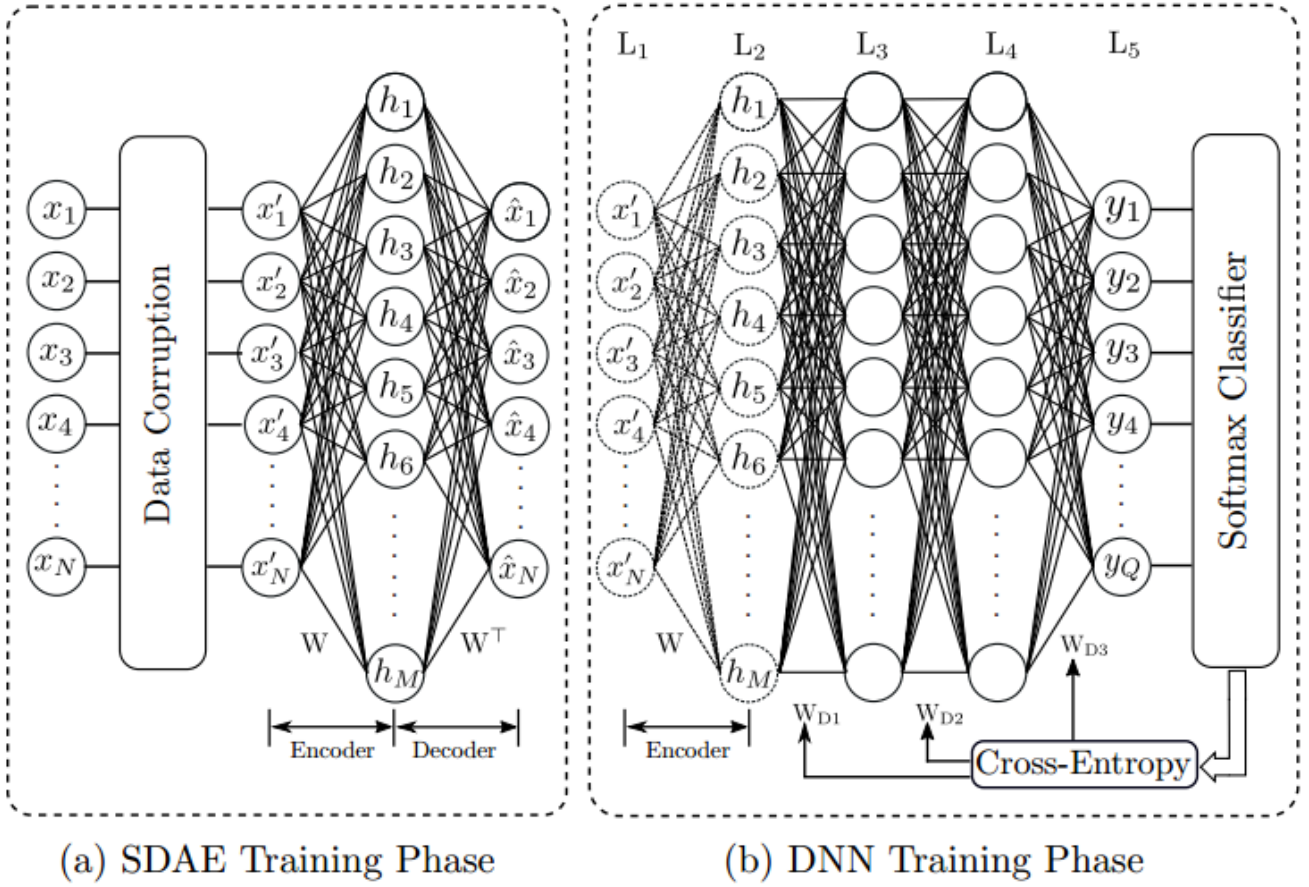
1. L2 usa a mesma função de ativação **f**.
2. As camadas ocultas L3 e L4 usam a **Unidade de revestimento retificado (ReLU)** como sua função de ativação.
3. Os dados **precisam ser rotulados nas classes Q** devido ao uso do classificador softmax, onde o rótulo é a direção do sinal do drone vindo.

TensorFlow

Validação experimental



Top view



(a) SDAE Training Phase (b) DNN Training Phase

L5- oito neurônios (temos **oito classes** para a classificação de direção).

L1- quatro neurônios (o conjunto de **antenas tem quatro elementos**, ou, N = 4).

L2, L3, L4- 200, 12, 12 e neurônios, respectivamente. Esses valores foram decididos **empiricamente** durante o processo de treinamento.

Considerações sobre código e alterações feitas

```python
if __name__ == "__main__":
    data = HandleData(total_data=880,data_per_angle=110,num_angles=8)
    antenna_data,label_data = data.get_synthatic_data(test_data=False)

    data_test = HandleData(total_data=80, data_per_angle=10, num_angles=8)
    antenna_data_test, label_data_test = data_test.get_synthatic_data(test_data=True)

def get_synthatic_data(self,test_data):

    if test_data is False:
        x_0 = genfromtxt('./Dround_Data_New/Nomalized/deg_0_normalize.csv', delimiter=',', dtype=np.float32)
        x_45 = genfromtxt('./Dround_Data_New/Nomalized/deg_45_normalize.csv',delimiter=',', dtype=np.float32)
        x_90 = genfromtxt('./Dround_Data_New/Nomalized/deg_90_normalize.csv', delimiter=',', dtype=np.float32)
        x_135 = genfromtxt('./Dround_Data_New/Nomalized/deg_135_normalize.csv',delimiter=',', dtype=np.float32)
        x_180 = genfromtxt('./Dround_Data_New/Nomalized/deg_180_normalize.csv', delimiter=',', dtype=np.float32)
        x_225 = genfromtxt('./Dround_Data_New/Nomalized/deg_225_normalize.csv',delimiter=',', dtype=np.float32)
        x_270 = genfromtxt('./Dround_Data_New/Nomalized/deg_270_normalize.csv', delimiter=',', dtype=np.float32)
        x_315 = genfromtxt('./Dround_Data_New/Nomalized/deg_315_normalize.csv',delimiter=',', dtype=np.float32)
    elif test_data is True:
        x_0 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_0_normalize.csv', delimiter=',', dtype=np.float32)
        x_45 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_45_normalize.csv',delimiter=',', dtype=np.float32)
        x_90 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_90_normalize.csv', delimiter=',', dtype=np.float32)
        x_135 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_135_normalize.csv',delimiter=',', dtype=np.float32)
        x_180 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_180_normalize.csv', delimiter=',', dtype=np.float32)
        x_225 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_225_normalize.csv',delimiter=',', dtype=np.float32)
        x_270 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_270_normalize.csv', delimiter=',', dtype=np.float32)
        x_315 = genfromtxt('./Dround_Data_New/Nomalized_test/deg_315_normalize.csv',delimiter=',', dtype=np.float32)
```

Os dados de treinamento e teste já estavam predefinidos e sequencialmente distribuidos.

→ TREINAMENTO

→ TESTE

Alterações

```python
if __name__ == '__main__':
    test_percentage = 0.2

    #instance of the Handle Data class
    data = HandleData(oneHotFlag=False)
    #get the data
    antenna_data, label_data = data.get_synthatic_data()
    antenna_data, antenna_data_test, label_data, label_test = train_test_split (antenna_data, label_data, \
        test_size=test_percentage, random_state=42)

def get_synthatic_data(self):
    x_0 = genfromtxt('./Dround_Data_New/Nomalized/deg_0_normalize.csv', delimiter=',', dtype=np.float32)
    x_45 = genfromtxt('./Dround_Data_New/Nomalized/deg_45_normalize.csv',delimiter=',', dtype=np.float32)
    x_90 = genfromtxt('./Dround_Data_New/Nomalized/deg_90_normalize.csv', delimiter=',', dtype=np.float32)
    x_135 = genfromtxt('./Dround_Data_New/Nomalized/deg_135_normalize.csv',delimiter=',', dtype=np.float32)
    x_180 = genfromtxt('./Dround_Data_New/Nomalized/deg_180_normalize.csv', delimiter=',', dtype=np.float32)
    x_225 = genfromtxt('./Dround_Data_New/Nomalized/deg_225_normalize.csv',delimiter=',', dtype=np.float32)
    x_270 = genfromtxt('./Dround_Data_New/Nomalized/deg_270_normalize.csv', delimiter=',', dtype=np.float32)
    x_315 = genfromtxt('./Dround_Data_New/Nomalized/deg_315_normalize.csv',delimiter=',', dtype=np.float32)
```

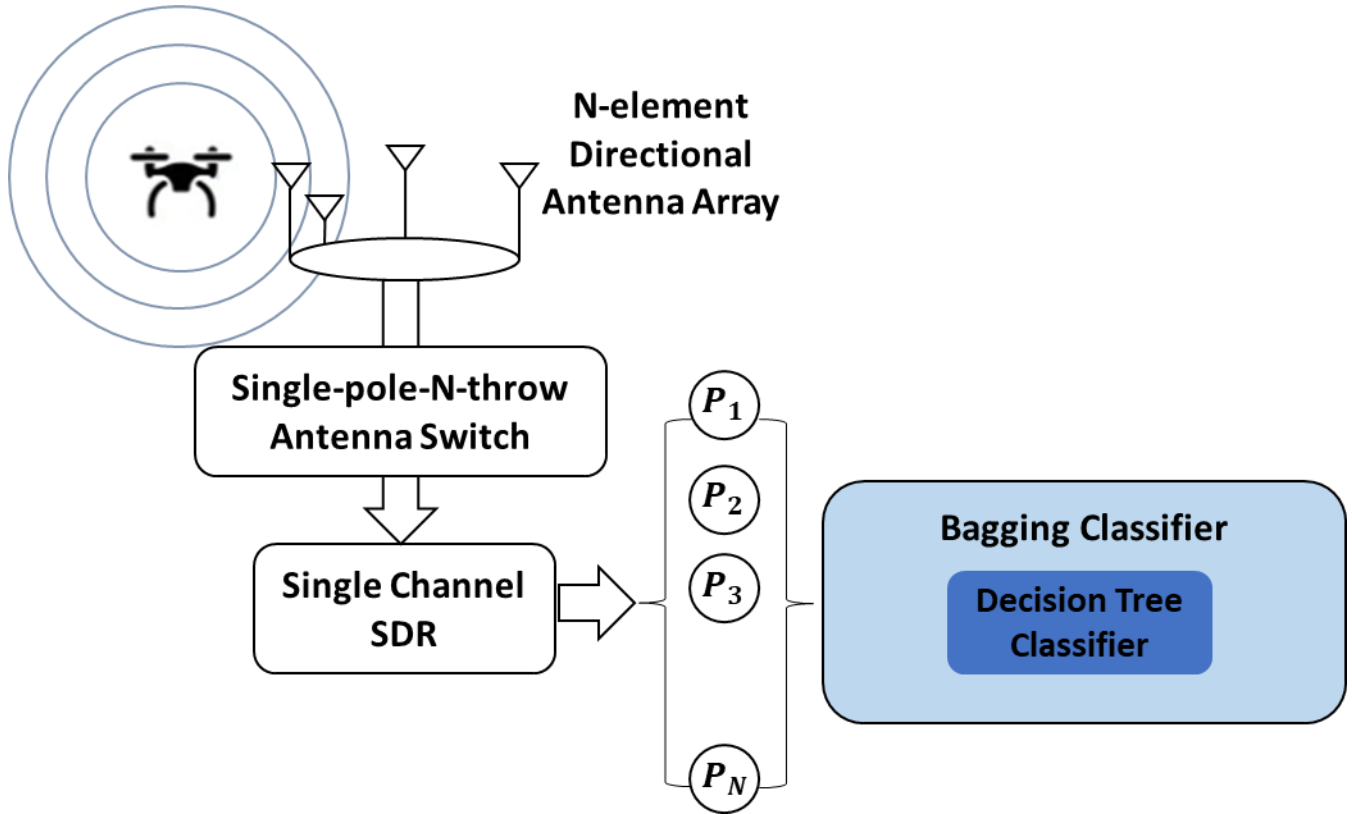
→ TREINAMENTO + TESTE

Outros modelos investigados

- **Árvore de decisão**.
- **Bagging Classifier.**

Outros modelos investigados

- **Árvore de decisão**.
- Bagging Classifier.



Bagging Classifier

```python
import tensorflow as tf
import math
import numpy as np
from get_csv_data import HandleData
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from time import time

if __name__ == '__main__':
    test_percentage = 0.2

    seed = 1234

    #instance of the Handle Data class
    data = HandleData(oneHotFlag=False)
    #get the data
    antenna_data, label_data = data.get_synthatic_data()
    antenna_data, antenna_data_test, label_data, label_test =
        test_size=test_percentage, random_state=42)

    # get denoising autoencoder outputs for the train and test data
    DAE_out = getDAE([antenna_data, antenna_data_test], seed)
    antenna_data = DAE_out[0]
    antenna_data_test = DAE_out[1]

    #Instantiate a Bagging Classifier
    clf = BaggingClassifier(DecisionTreeClassifier(random_state=42), n_estimators=300, max_samples=250, \
        bootstrap=False, n_jobs=-1, random_state=42)
    #Train model
    tic = time()
    clf.fit(antenna_data, label_data)
    #Predict
    y_pred = clf.predict(antenna_data_test)
```

```python
class HandleData(object):

    def __init__(self, oneHotFlag=False):
        self.total_data = genfromtxt('./Dround_Data_New/Nomalized/deg_0_normalize.csv', delimiter=',', \
            dtype=np.float32).shape[0]*len(os.listdir('./Dround_Data_New/Nomalized'))
        self.data_per_angle = genfromtxt('./Dround_Data_New/Nomalized/deg_0_normalize.csv', delimiter=',', \
            dtype=np.float32).shape[0]
        self.num_angles = len(os.listdir('./Dround_Data_New/Nomalized'))
        self.current_point = 0
        self.data_set = np.zeros((self.total_data, 4), dtype=np.float32)
        if oneHotFlag == True:
            self.label_set = np.zeros((self.total_data, self.num_angles), dtype=np.float32)
        else:
            self.label_set = [0 for i in range(self.total_data)]
        self.oneHotFlag = oneHotFlag


        for i in range(0, self.num_angles):
            for j in range(0, self.data_per_angle):
                "add one hot"
                "add data"
                if self.oneHotFlag == True:
                    self.label_set[i * self.data_per_angle + j] = self.onehot_encode(i)
                else:
                    self.label_set[i * self.data_per_angle + j] = i
                self.data_set[i * self.data_per_angle + j] = data_matrix[i][j]

        return self.data_set ,self.label_set
```

0^0	45^0	90^0	135^0	180^0	225^0	270^0	315^0
0	1	2	3	4	5	6	7

Treina um conjunto de 300 classificadores da Árvore de Decisão.

Cada um treinado em 250 instâncias de treinamento amostradas aleatoriamente do conjunto de treinamento.

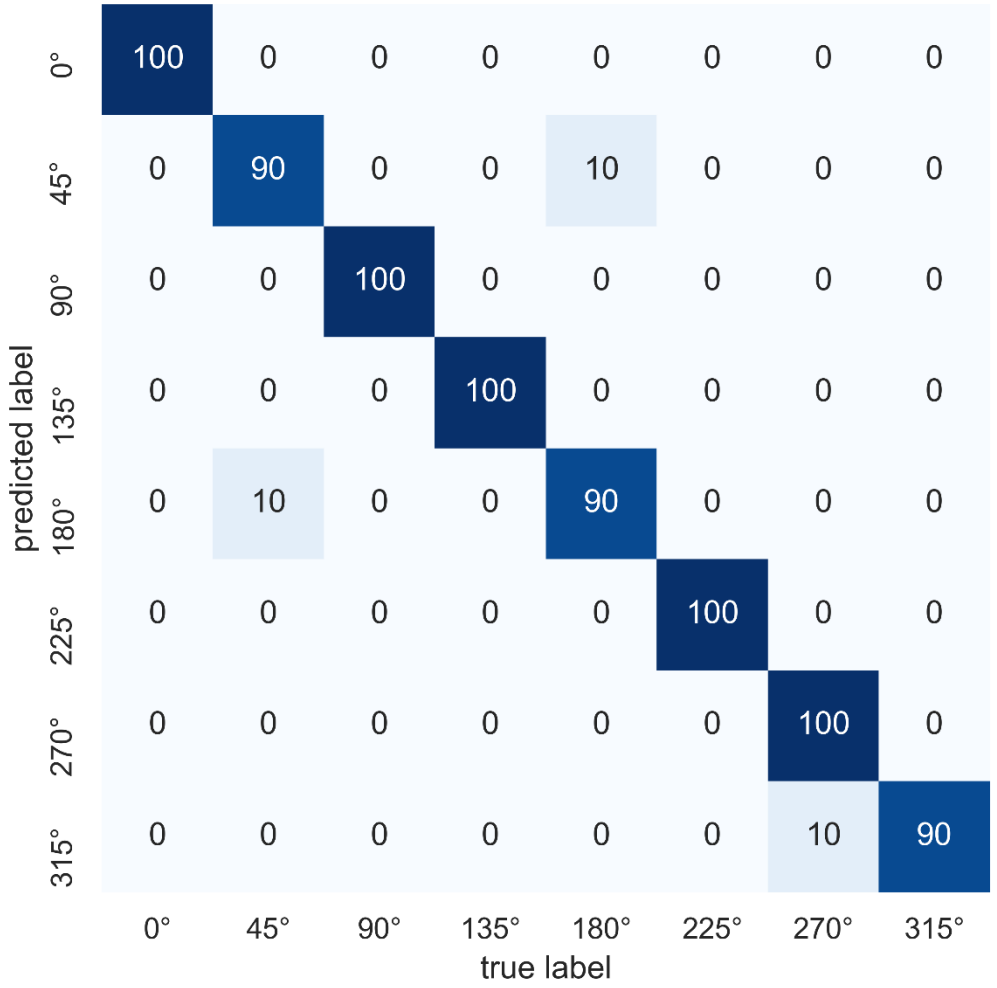
A amostragem sem substituição é realizada.

–1 diz ao Scikit-Learn para usar todos os núcleos do CPU disponíveis.

Comparação dos métodos (8 % dos dados para teste)

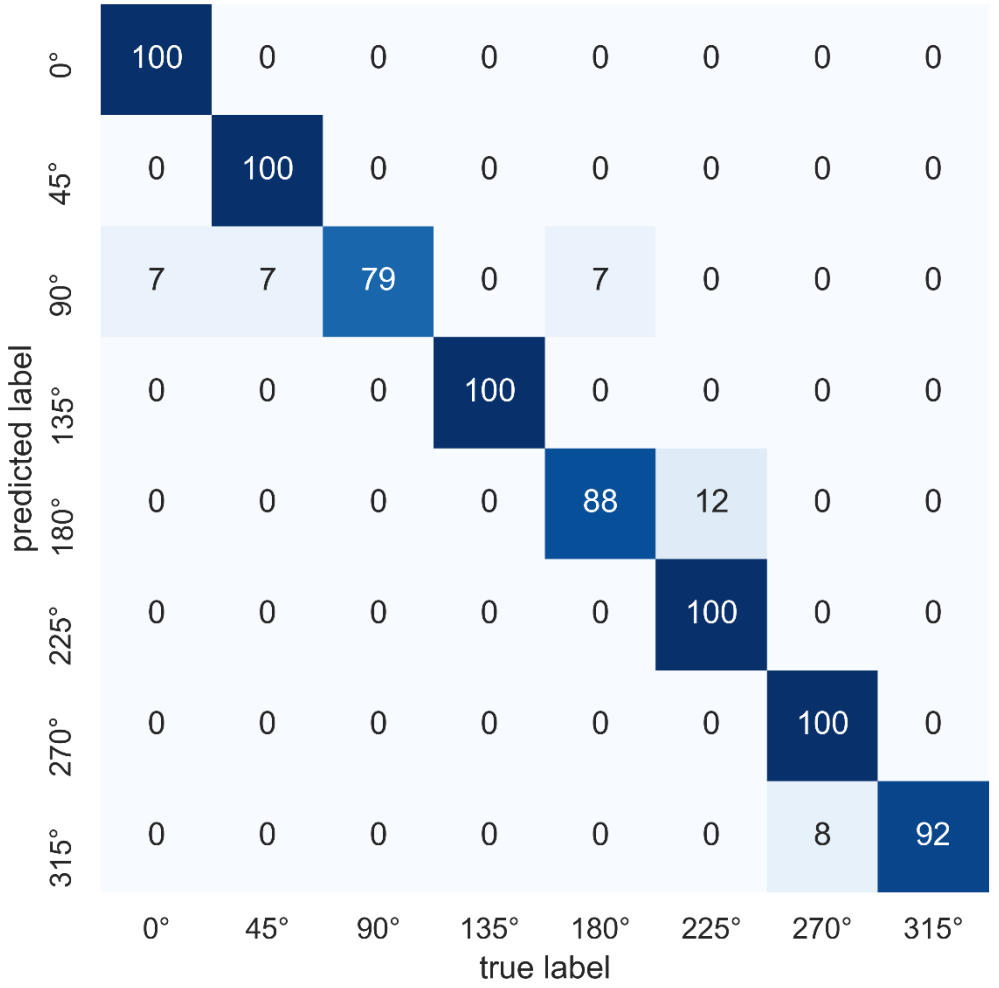
DNN sim modificações

```
Epoch: 2000 cost= 0.119282786
Optimization Finished!
Elapsed time: 109.61747765541077 seconds
Accuracy: 0.96022725
Accuracy: 0.9625
```



Bagging Classifier

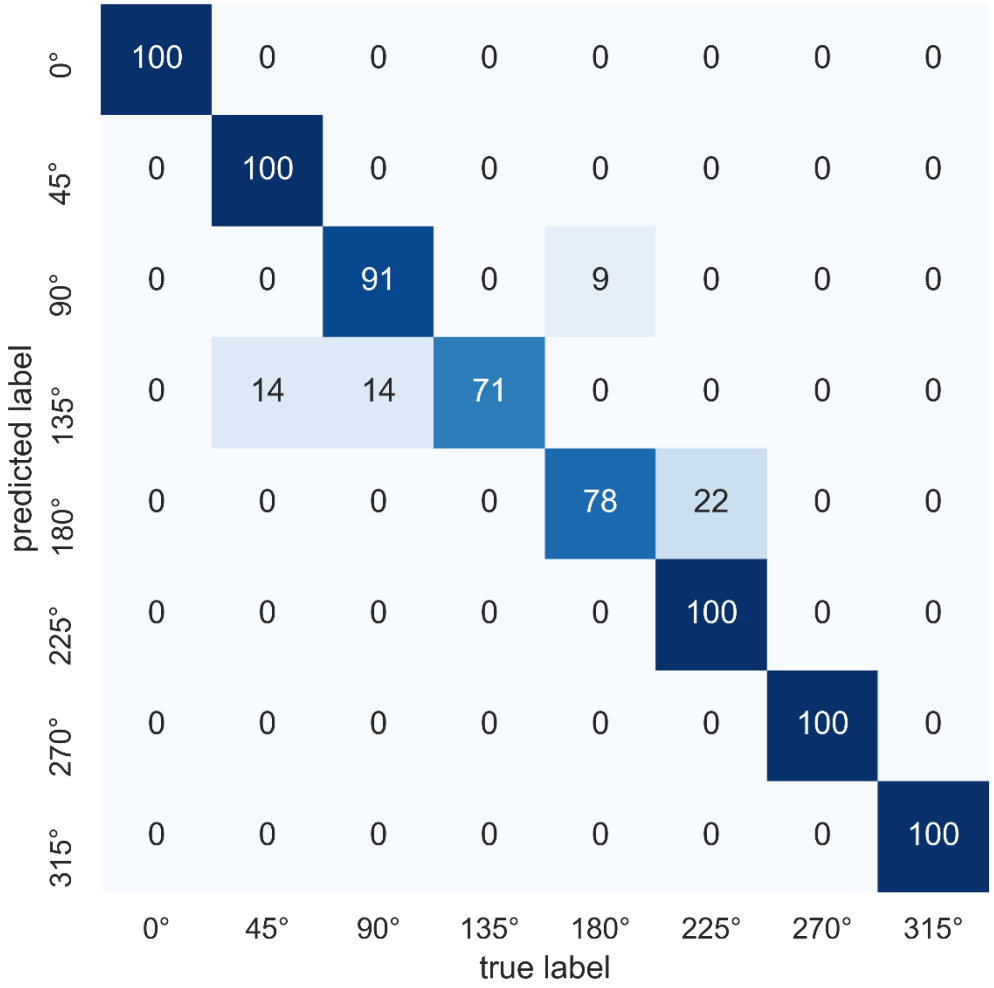
```
Accuracy of model is: 93.5064935064935 %
Elapsed time: 1.4059202671051025 seconds
```



Comparação dos métodos (8 % dos dados para teste)

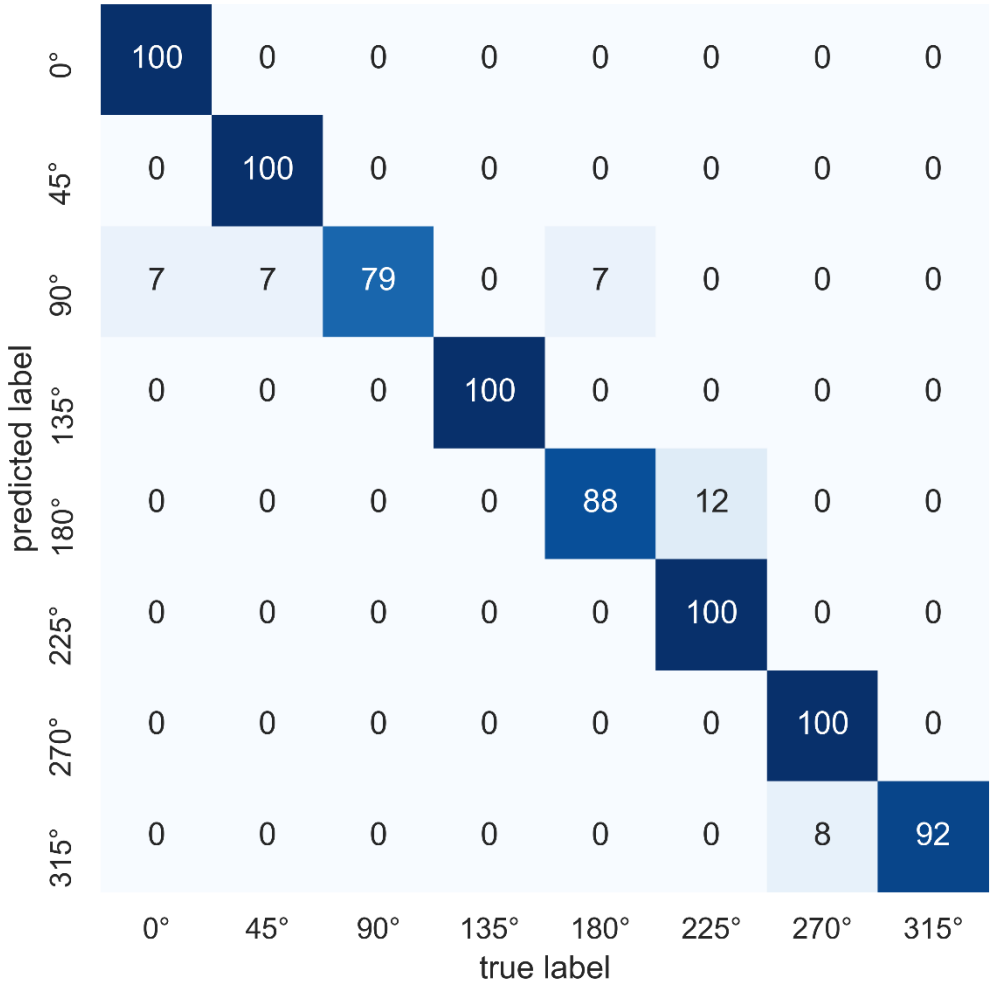
DNN com modificações

```
Epoch: 2000 cost= 0.164094241
Optimization Finished!
Elapsed time: 109.2904200553894 seconds
Accuracy (train data): 0.9535674
Accuracy (test data): 0.9350649
```



Bagging Classifier

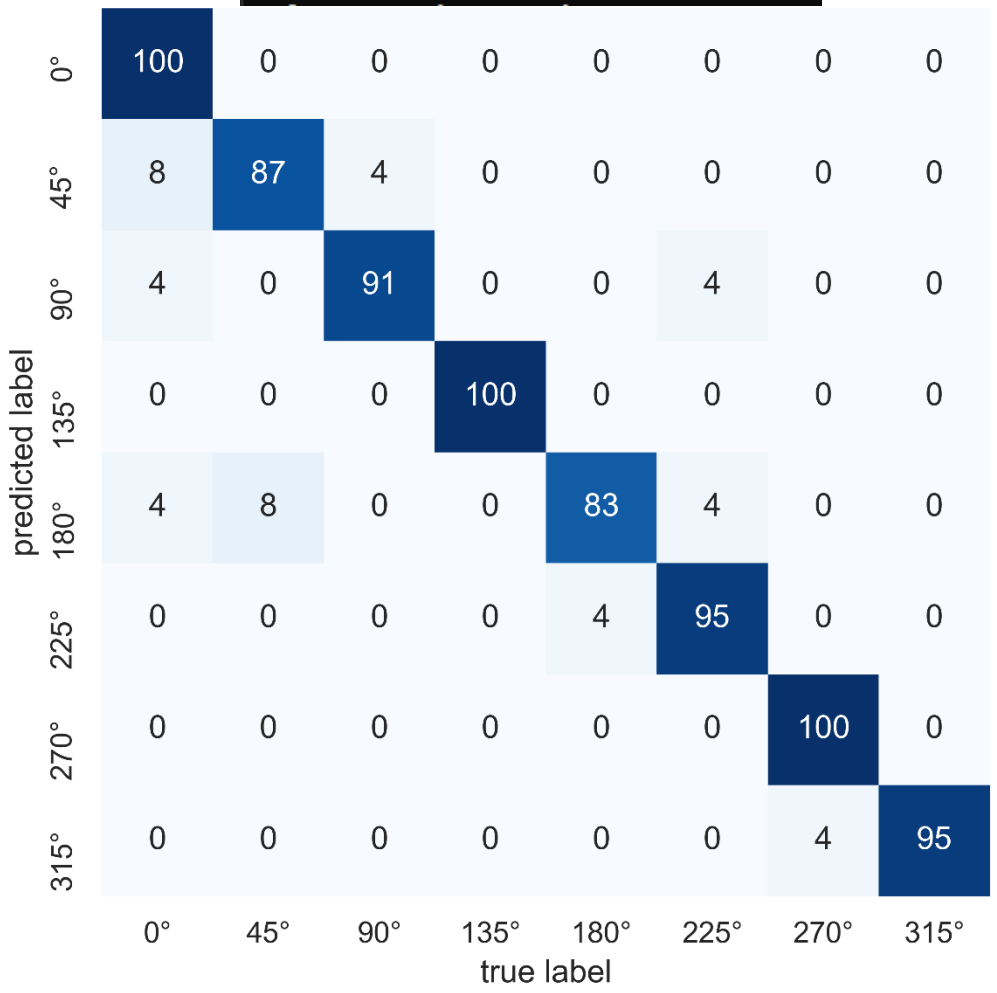
```
Accuracy of model is: 93.5064935064935 %
Elapsed time: 1.4059202671051025 seconds
```



Comparação dos métodos (20 % dos dados para teste)

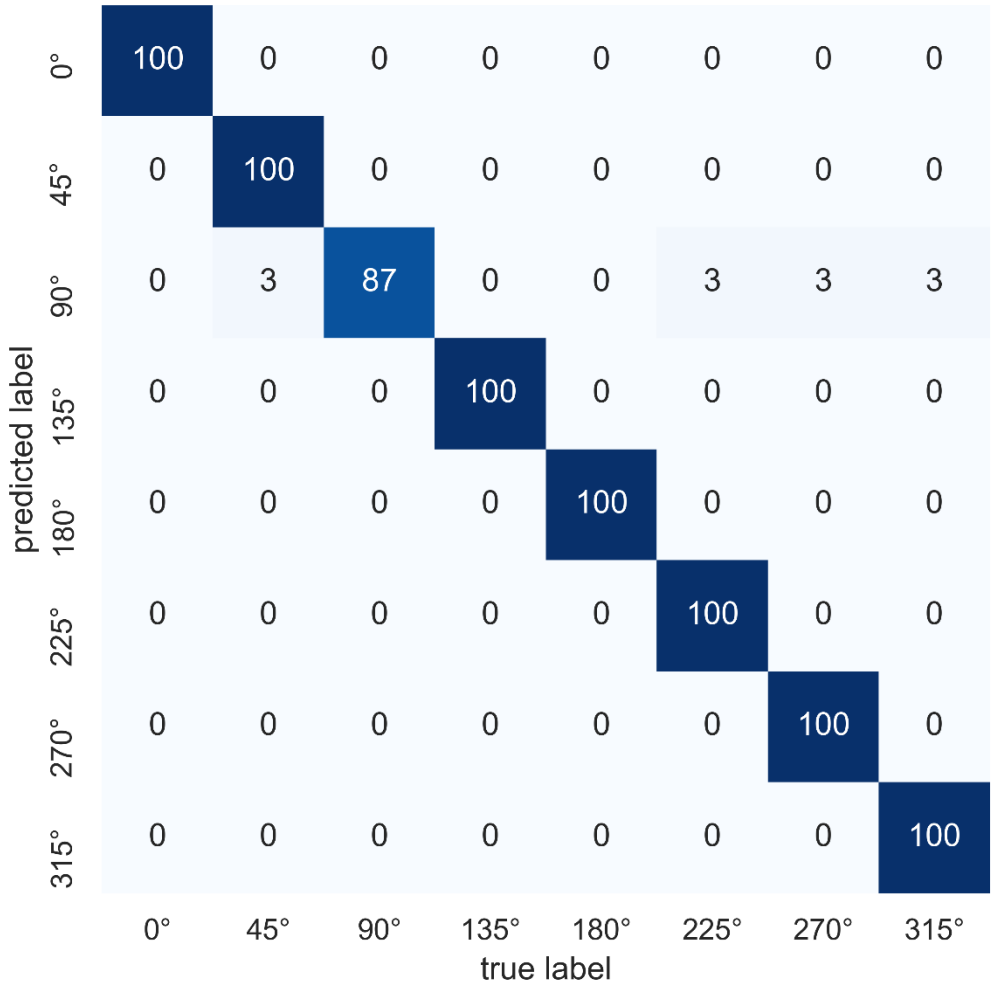
DNN sim modificações

```
Epoch: 2000 cost= 0.153396163
Optimization Finished!
Elapsed time: 96.26545906066895 seconds
Accuracy (train data): 0.95394737
Accuracy (test data): 0.953125
```



Bagging Classifier

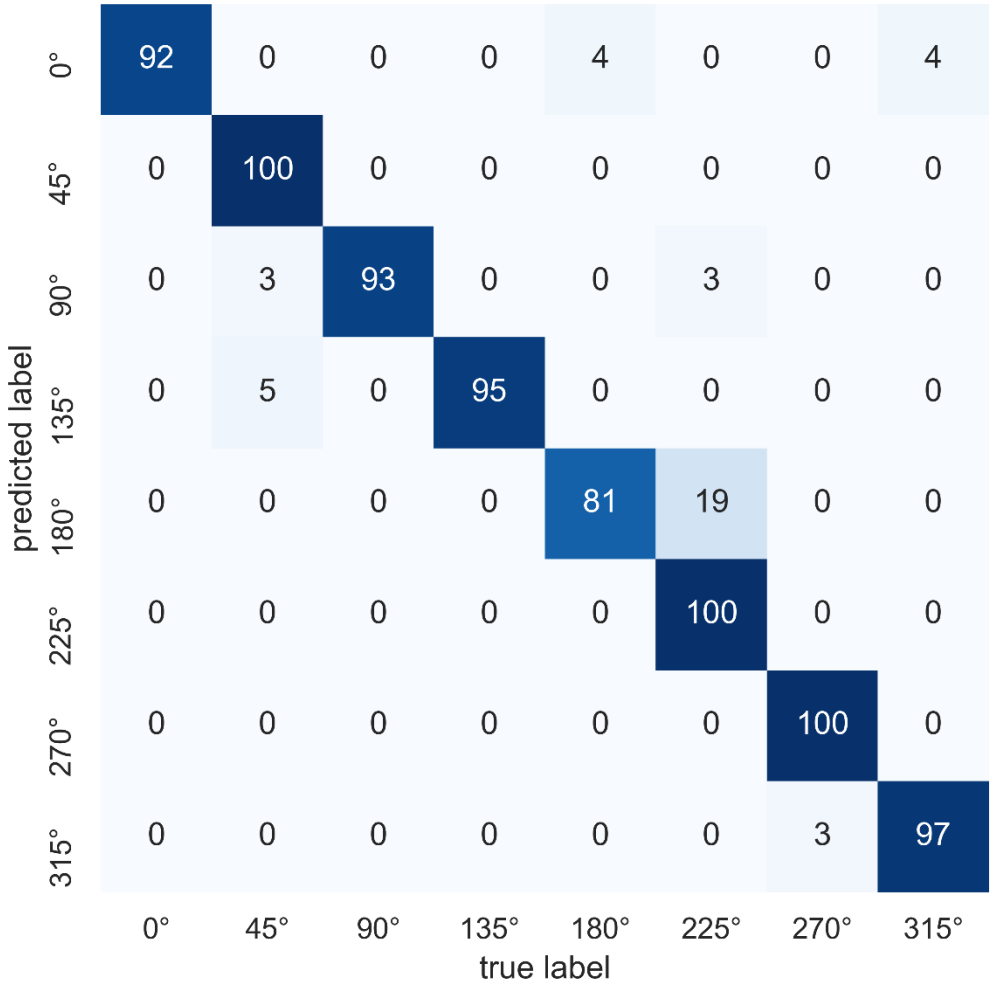
```
Accuracy of model is: 97.91666666666666 %
Elapsed time: 1.4230878353118896 seconds
```



Comparação dos métodos (20 % dos dados para teste)

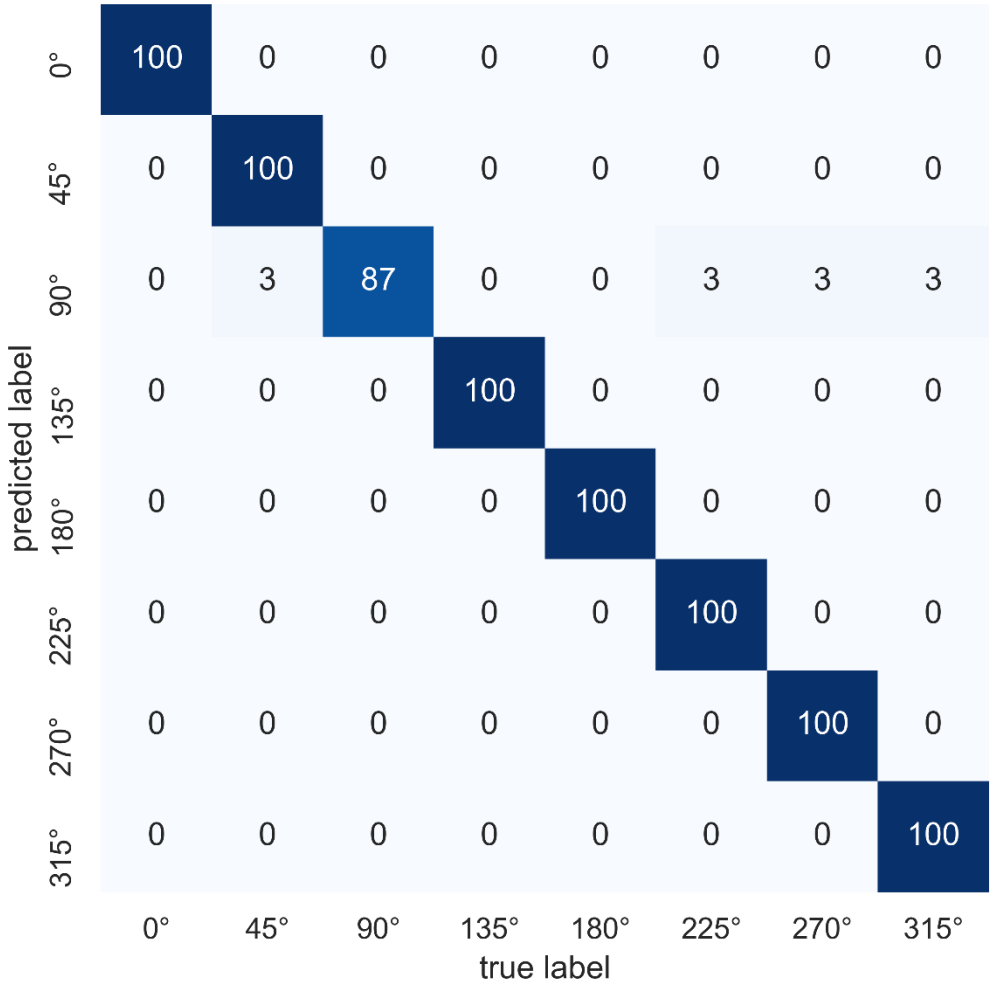
DNN com modificações

```
Epoch: 2000 cost= 0.174811864
Optimization Finished!
Elapsed time: 95.31753039360046 seconds
Accuracy (train data): 0.9557292
Accuracy (test data): 0.9322917
```



Bagging Classifier

```
Accuracy of model is: 97.91666666666666 %
Elapsed time: 1.4230878353118896 seconds
```



Conclusões

- É proposto um novo método DF para ser usado em um sistema de vigilância por drones.
- O sistema compreende um receptor de canal único e uma antena direcional.
- Se obteve uma relação subjacente entre a potência média do sinal recebido no conjunto da n-ésima antena e $\boldsymbol{\theta}$.
- Os dados de treinamento e teste estavam predefinidos e sequencialmente distribuidos.
- Apenas 8% dos dados foram utilizados para validação.
- São propostas melhorias no modelo DNN e além é proposto um modelo com melhor desempenho.

Trabalhos futuros

- Sistema inteligente capaz de identificar o estado de drone de maneira precisa.
- Propor um sistema com um arreglo de antennas capaz de detectar o angulo de elevação, além do angulo azimute.
- Um modelo de sistema com múltiplos canais.
- Aplicação das técnicas de múltiplo acesso não ortogonal para a detecção de drone.