

POLITECNICO
MILANO 1863

Deep Learning for Fault Diagnostic – Exercise Session

Ibrahim Ahmed

Politecnico di Milano, Italy

ibrahim.ahmed@polimi.it

Schedule

1, **Demo case:** CNN for Classification of MNIST Handwritten digits
(0-30 mins)

2, **Real case (Classwork):** CNN for Classification of Defect Patterns
on Semiconductor Wafers
(30-60 mins)

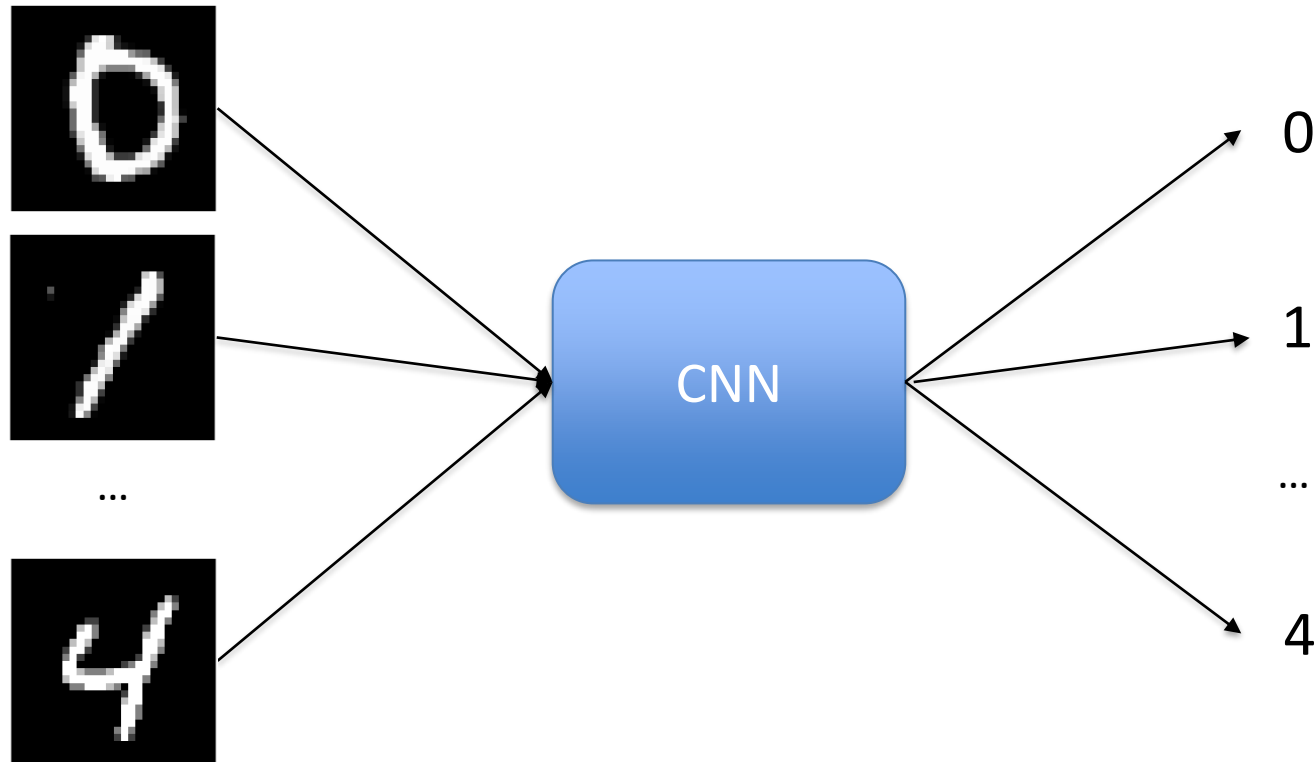
3, **Solution discussion for real case**
(60-90 mins)

CNN for Classification of MNIST Handwritten digits

Demo case

Simplified case:

- Only digits 0, 1, 2, 3, 4 considered



- Loading the data:

```
load('mnist.mat')
```

- Number of samples:

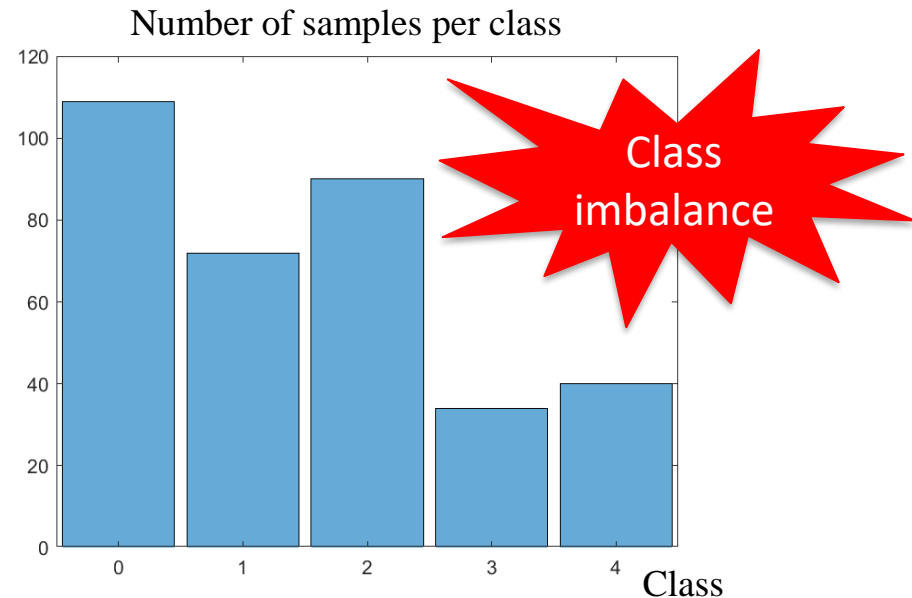
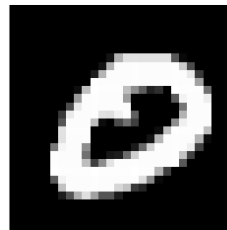
- Training set: 345
- Validation set: 172
- Test set: 173

- Data distribution in training set:

```
histogram(y_train)
```

- Visualizing one data sample:

```
imshow(x_train(:, :, :, 1))
```



Data preprocessing

- Training set → used for training the model
- Validation set → used for monitoring training to prevent overfitting
- Test set → used for testing the developed model

- Data normalization:

➤ The pixel values are between 0 and 255.

```
x_train = x_train ./ 255;
```

```
x_valid = x_valid ./ 255;
```

```
x_test = x_test ./ 255;
```

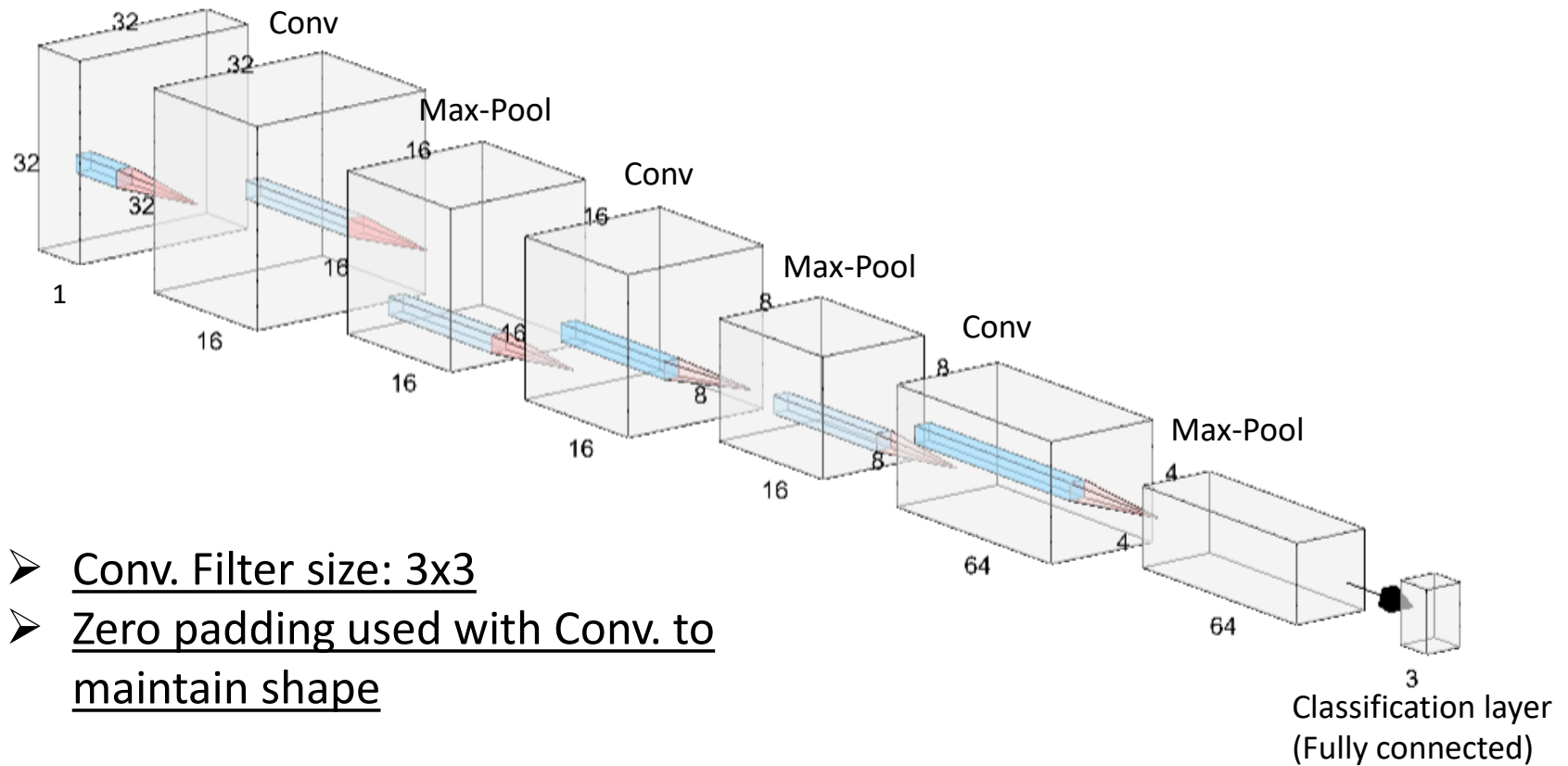
- Changing the labels to categorical type:

```
y_train = categorical(y_train);
```

```
y_valid = categorical(y_valid);
```

```
y_test = categorical(y_test);
```

CNN Architecture



Defining the model

- Defining the layers:

```
layers = [  
    imageInputLayer([28 28 1])  
    convolution2dLayer(3, 16, 'Padding', 1)  
    reluLayer  
    maxPooling2dLayer(2, 'Stride', 2)  
    convolution2dLayer(3, 16, 'Padding', 1)  
    reluLayer  
    maxPooling2dLayer(2, 'Stride', 2)  
    convolution2dLayer(3, 64, 'Padding', 1)  
    reluLayer  
    maxPooling2dLayer(2, 'Stride', 2)  
    fullyConnectedLayer(5)  
    softmaxLayer  
    classificationLayer];
```

Activation
function

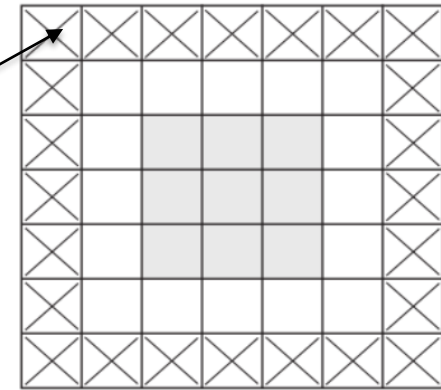
Input images dimension

Number of filters

Filter size

Number of pixels by which the
filter is shifted

Number of
neurons



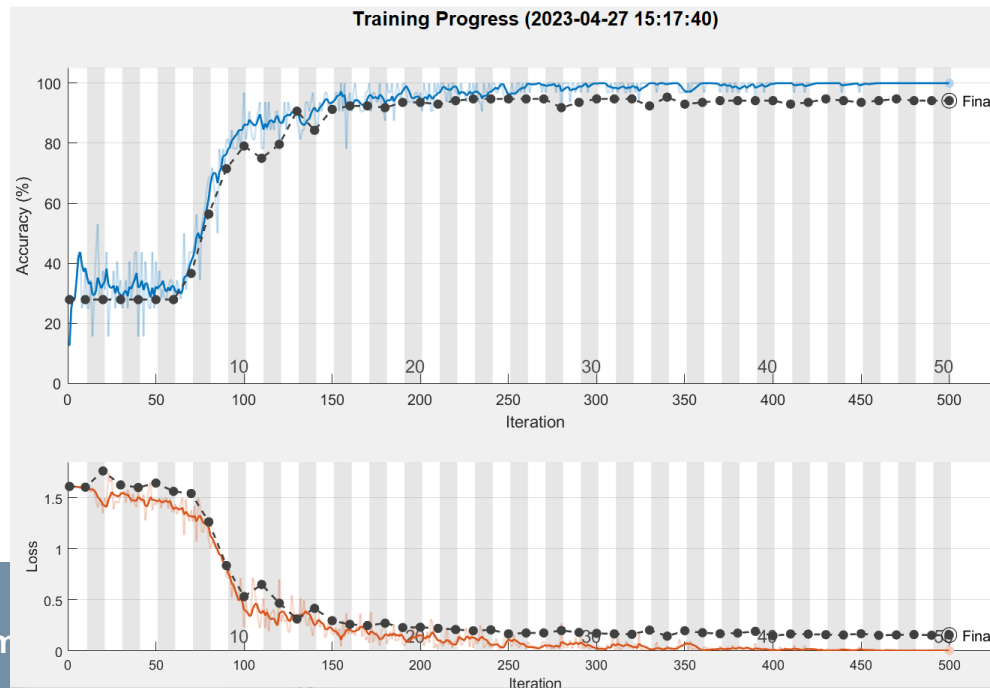
Training the model

```
miniBatchSize = 32;
options = trainingOptions( 'adam', 'MiniBatchSize', miniBatchSize, ...
    'MaxEpochs', 50, 'Shuffle', 'every-epoch', ...
    'ValidationData', {x_valid, y_valid}, 'ValidationFrequency', 10, ...
    'Plots', 'training-progress');

net = trainNetwork(x_train, y_train, layers, options);
```

Optimizer

Validation is done every 10 iterations



Testing the model

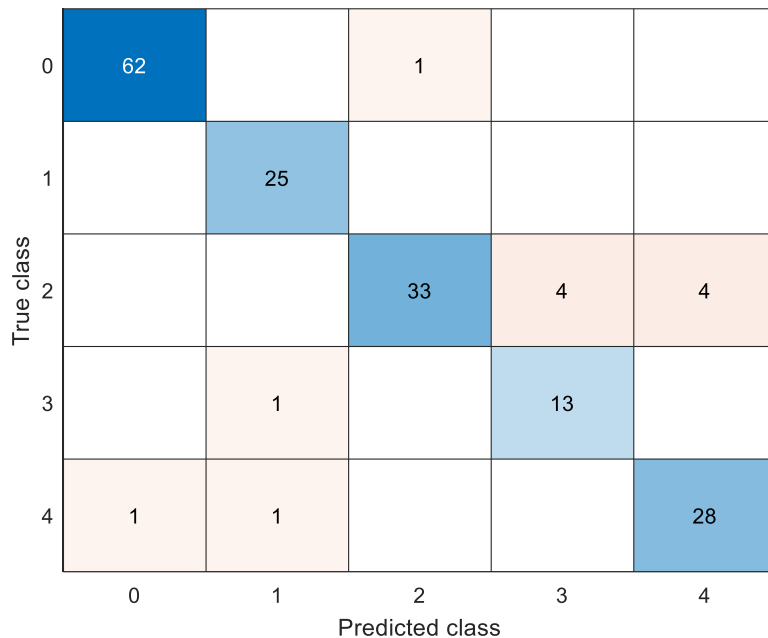
```
y_pred = classify(net, x_test);  
test_accuracy = sum(y_pred == y_test) / numel(y_test);
```

❖ Test set accuracy = 93.60%

➤ Confusion matrix:

```
confusionchart(y_test, y_pred)
```

Underperformance
in
underrepresented
classes



Data augmentation

- ❖ We need to increase the number of samples in classes with fewer samples by creating artificial data (oversampling)
- We can randomly select some data from an underrepresented class, rotate them, add noise to them, etc., and add them to the original samples in that class

```
img = squeeze(x_train(:, :, :, 8));  
imshow(img)
```



- Rotating:

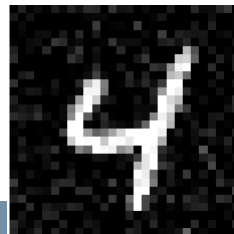
```
imshow(rotate(img))
```



Rotating is not a good idea in this case

- Adding noise:

```
imshow(add_noise(img))
```



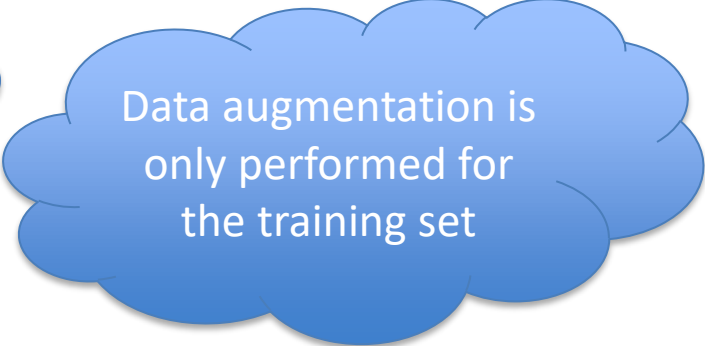
Functions

Data augmentation

```
[n_per_class, edges] = histcounts(y_train);
classes = [0 1 2 3 4];
max_class = max(n_per_class);
add_per_class = max_class - n_per_class;

idx_added = [];
for c = 1:5
    added = 0;
    while added < add_per_class(c)
        idx_add = randi(size(y_train, 1));
        if y_train(idx_add) == categorical(classes(c))
            idx_added(end + 1) = idx_add;
            added = added + 1;
        end
    end
end

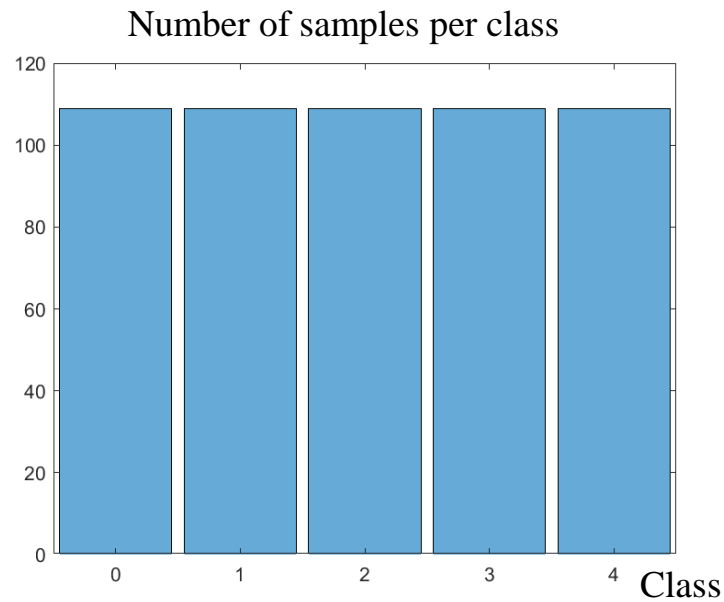
x_train_added = add_noise(x_train(:, :, :), idx_added);
y_train_added = y_train(idx_added);
x_train_aug = cat(4, x_train, x_train_added);
y_train_aug = cat(1, y_train, y_train_added);
```



Data augmentation is
only performed for
the training set

Data augmentation

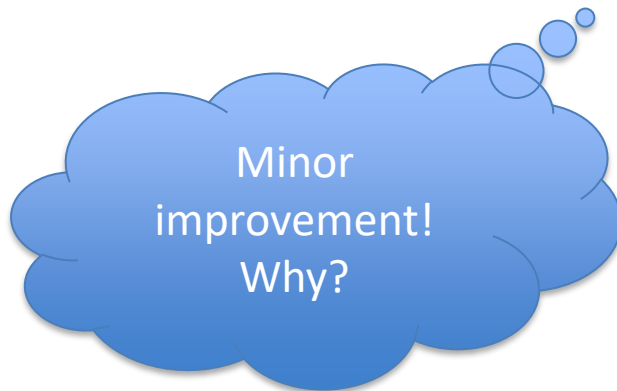
❖ Training data class distribution after class balancing:



Training the network with the augmented dataset

❖ Test set accuracy: 95.38%

❖ Confusion matrix:



True class	0	1	2	3	4
0	63				
1		25			
2	1		37	2	1
3				14	
4	1	2	1		26
		Predicted class			

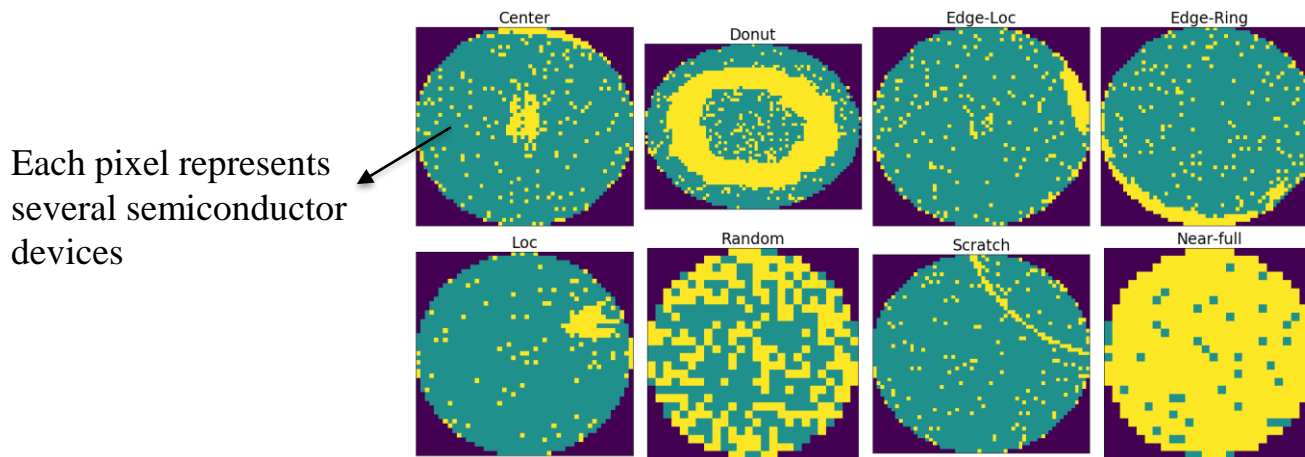
CNN for Classification of Defect Patterns on Semiconductor Wafers

Case Study

WM-811k Wafer Map Dataset

Maps of device failure on semiconductor wafers

Each specific failure pattern is related to a specific known defect in the production line



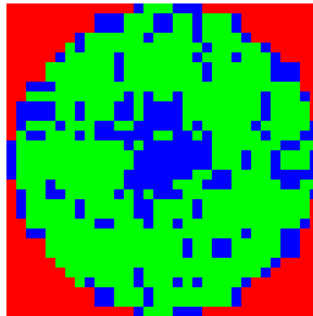
❖ Problem: Automatic classification of failure patterns

➤ Solution: A CNN classifier

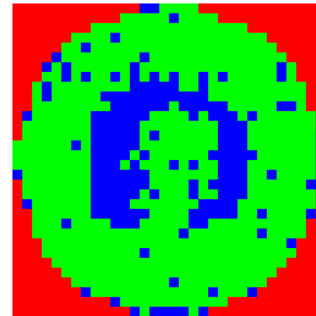
Dataset

❖ Only 3 classes considered:

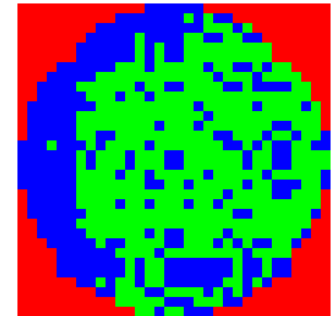
- Center
- Donut
- Edge-Loc



Center



Donut



Edge-Loc

❖ Number of samples in each class:

- Center: 4294
- Donut: 555
- Edge-Loc: 5189



Dataset

```
imshow(center(:, :, :, 2))
```



0 (Healthy/Background)

1 (Defective)

Exercise

1. Divide the data into training, validation, and test sets, taking the same percentage of data from each class (E.g., 50% per class for training set, etc.)
2. Train a CNN (same architecture used in the previous case study) on this dataset and discuss the classification accuracy.
3. Plot the classification accuracy vs. different percentages of data per class taken for training (5%, 10%, 15%, ..., 80%). Divide the rest of the data equally in validation and test sets. What is the optimum number of training samples for this problem?
4. Repeat the previous exercise, but this time also performing data augmentation for balancing the classes. Rotate and add noise to randomly selected samples from each class in the training set.



Thanks for the attention