**Eskişehir Osmangazi Üniversitesi**

**Neural Networks**

**152118003**

**Fatih Özcan**

**152120211153**

**Murat Özcan**

**152120221097**

**Assoc. Prof. Eyüp Çinar**

**2024-2025**

## Table of Contents

## Video Link:

https://drive.google.com/drive/folders/1sniyBXAKSfPNCzWyZ_beMC2_-U2u6iEM?usp=sharing

## Github Link:

fatihozcann/Neural-Networks: Neural Networks Final Project

### Model 1: AutoEncoder

#### Model Overview

The AutoEncoder is an unsupervised neural network model designed to detect anomalies by learning to reconstruct "normal" wood surface images. It is trained solely on defect-free images and learns to recreate them as accurately as possible. During testing, images with defects are expected to have larger reconstruction errors, signaling anomalies.

#### Architecture and Methodology

- **Architecture:**
  The model uses a convolutional autoencoder with three encoding layers and three decoding layers. The encoder progressively downsamples the input image (256x256 grayscale) via convolutional and max-pooling layers to a compressed latent representation. The decoder upsamples this representation through convolutional and upsampling layers to reconstruct the original image.

- **Layers:**

  - Encoder: Conv2D (32 filters) → MaxPooling → Conv2D (64 filters) → MaxPooling → Conv2D (128 filters) → MaxPooling

  - Decoder: Conv2D (128 filters) → UpSampling → Conv2D (64 filters) → UpSampling → Conv2D (32 filters) → UpSampling → Conv2D (1 filter with sigmoid activation)

- **Loss:** Mean Squared Error (MSE) between original and reconstructed images.

- **Input:** 256x256 grayscale images, normalized between 0 and 1.

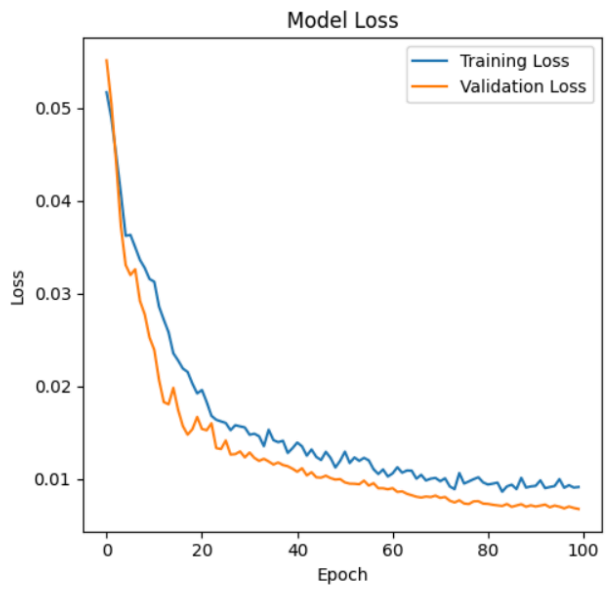
- **Data Augmentation:** Rotation, width/height shifts, and horizontal flips applied to training images to enhance model robustness.

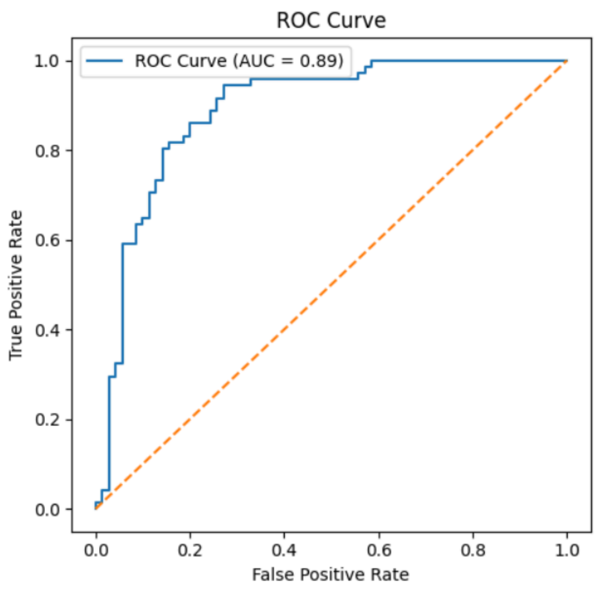
### Training Details

- **Dataset:**
    - Training on defect-free (good) images only.
    - Validation split: 10% of the training data.
    - Test set contains both good and defective images.
- **Optimization:** Adam optimizer with learning rate = 1e-4.
- **Epochs:** 100.
- **Batch size:** 16.
- **Device:** Trained on T4 GPU via Google Colab.

### Performance Metrics

- **Results:**
    - F1 Score: 0.84
    - ROC AUC: 0.88
    - Optimal Threshold: 0.18





## Model 2: EfficientAD

### Model Overview

EfficientAD is an anomaly detection model leveraging knowledge distillation between a pretrained EfficientNet-B0 (teacher) and a lightweight CNN (student). The model detects anomalies by learning feature representations from normal images and measuring discrepancies between teacher and student features, combined with reconstruction error to identify anomalies in wood surface images.

### Architecture and Methodology

- **Backbone (Teacher):**
  Uses EfficientNet-B0 pretrained on ImageNet to extract multi-level features from input images. The teacher network parameters are frozen during training.

- **Student Network:**
  A compact CNN with four convolutional blocks that extracts features at multiple levels designed to mimic the teacher's feature space.

- **Feature Projection:**
  Both teacher and student features are projected into a common 64-channel space using 1x1 convolutions for effective comparison.

- **Decoder:**
  The student's mid-level features are decoded via transposed convolutions to

reconstruct the input image.

- **Anomaly Scoring:**
  Combines three components: MSE loss between student and teacher early features, mid-level features, and reconstruction error. These are spatially aligned and weighted to generate an anomaly heatmap and an image-level anomaly score.

**Training Details**

- **Dataset:**
  Training uses only defect-free (good) wood images.
  A 90/10 train-validation split is used for training monitoring.

- **Optimization:**
  Adam optimizer with a learning rate of 1e-4.
  Cosine Annealing learning rate scheduler over 20 epochs.

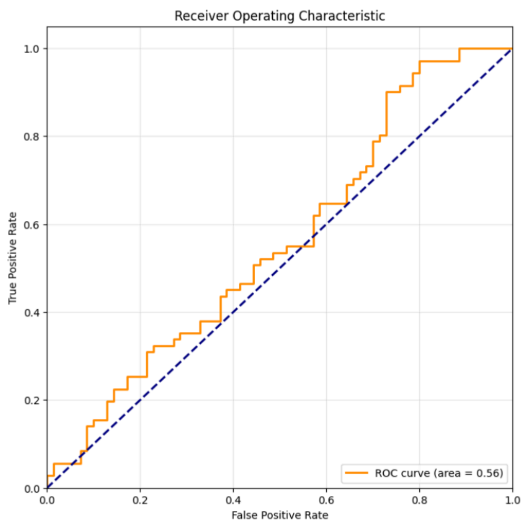
- **Loss Function:**
  Total loss is a weighted sum of feature distillation loss (between student and teacher features) and reconstruction loss, balanced with uncertainty weighting (learned log variance).

- **Batch Size:** 16.

- **Device:** Trained on T4 GPU via Google Colab.

**Performance Metrics**

- **Results:**
  - F1 Score: 0.68
  - ROC AUC: 0.55
  - Optimal Threshold: 51.09



## Model 3: GANomaly (GAN-based Anomaly Detection)

### Model Overview

GANomaly is an adversarial anomaly detection model designed to identify anomalies by learning to reconstruct normal images using a generator-discriminator setup. The generator has an encoder-decoder-encoder structure, encoding the input image into a latent space, reconstructing it, then re-encoding the reconstruction to enforce consistency. Anomalies are detected via the reconstruction error and latent space discrepancies.

### Architecture and Methodology

- **Generator:**
  - Encoder-Decoder-Encoder architecture with convolutional layers and LeakyReLU activations.
  - First encoder compresses the input image into a latent representation.
  - Decoder reconstructs the image from the latent vector.
  - Second encoder re-encodes the reconstructed image to a latent vector, encouraging latent space consistency.
- **Discriminator:**
  - Standard CNN classifier distinguishing real images from generated images.
  - Includes convolutional layers followed by dense layers and a sigmoid output for real/fake classification.

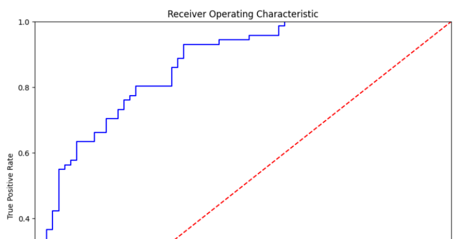
- **Loss Functions:**
    - **Adversarial loss:** Binary cross-entropy to fool the discriminator.
    - **Contextual loss:** Mean squared error (MSE) between input and reconstructed images.
    - **Latent loss:** MSE between latent vectors of input and reconstruction to enforce latent consistency.
- **Training:**
  The generator and discriminator are trained adversarially with weighted loss components focusing on reconstruction quality and latent space alignment.

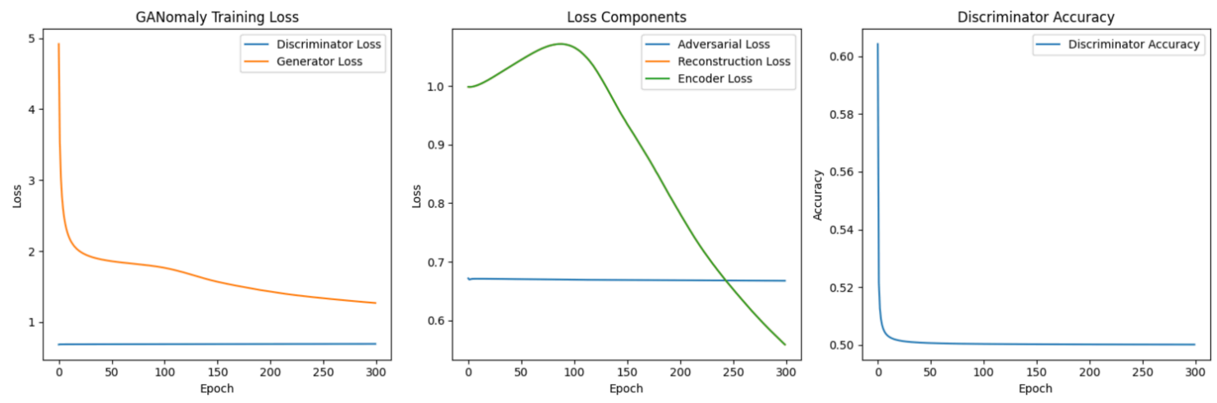
**Training Details**

- **Dataset:**
  Trained only on defect-free (good) images.
  Validation data split implicitly by monitoring discriminator and generator loss over epochs.
- **Optimization:**
  Adam optimizer with learning rate 0.0002 and beta_1=0.5.
- **Epochs:** 300 epochs with batch size 16.
- **Callbacks:** Visual progress monitoring every 5 epochs through image reconstruction visualization.
- **Model Saving:** Both generator and discriminator models saved after training.

**Performance Metrics**

- **Results:**
    - F1 Score: 0.8
    - ROC AUC:0.86
    - Optimal Threshold: 0.05





## Model 4: PADIM (Patch Distribution Modeling)

### Model Overview

PADIM is an anomaly detection approach based on modeling the distribution of multi-scale patch features extracted from a pretrained CNN backbone (ResNet18). It detects anomalies by computing the Mahalanobis distance between extracted patch features of test images and the learned multivariate Gaussian distribution of normal (defect-free) patches, enabling fine-grained detection of localized defects in wood images.

### Architecture and Methodology

- **Feature Extraction:**
  Uses a pretrained ResNet18 backbone modified for grayscale images, extracting features from three layers (layer1, layer2, layer3) to capture multi-scale information. Features are spatially resized and concatenated to form a rich representation for each patch.

- **Patch Distribution Modeling:**

  - For each spatial patch location, a multivariate Gaussian distribution (mean and covariance) is fit using patch features from normal training samples.

  - Dimensionality reduction via random projection selects a subset of feature channels for computational efficiency.

- **Anomaly Scoring:**
  The Mahalanobis distance is computed for each patch in a test image against the corresponding Gaussian distribution. The anomaly map is smoothed with a Gaussian filter. Image-level anomaly scores are the maximum patch distance values.

### Training Details

- **Dataset:**
  Training only on defect-free wood images.

- **Backbone:**
  Pretrained ResNet18 with modified first convolution layer to accept grayscale input.

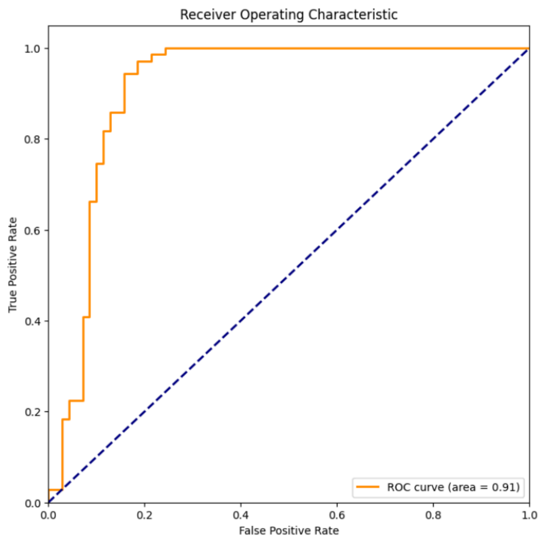
- **Dimension Reduction:**
  Random selection of feature channels (e.g., reduced dimension d=100) to reduce computational load during covariance estimation.

- **Covariance Calculation:**
  Covariance matrices inverted patch-wise with numerical stability adjustments (epsilon diagonal).

- **Device:**
  Computations performed on GPU if available.

### Performance Metrics

- **Results:**
  - F1 Score: 0.89
  - ROC AUC: 0.91



## Model 5: PatchCore

### Model Overview

PatchCore is a state-of-the-art anomaly detection method based on patch-level feature extraction and nearest neighbor search. It leverages pretrained CNN features and a memory bank of “normal” patch embeddings, performing anomaly detection by measuring the distance from test patches to their nearest neighbors in this memory bank.

**Architecture and Methodology**

- **Feature Extraction:**
  Uses a pretrained backbone CNN (in our case we used WideResNet50_2) modified for grayscale images. Features are extracted from multiple intermediate layers (e.g., layer2 and layer3) to capture multi-scale information.

- **Patch Features and Memory Bank:**

  - Extracts patch features by spatially reshaping feature maps and concatenating features from multiple layers at each patch location.

  - Uses **coreset sampling** (greedy farthest point sampling) to reduce the memory bank size, keeping a representative subset of normal patch features for efficiency.

- **Anomaly Scoring:**

  - At test time, nearest neighbor search is performed using the FAISS library to find distances between test patch features and the memory bank.

  - Patch-level distances are aggregated into anomaly maps, smoothed with a Gaussian filter.

  - Image-level anomaly scores are taken as the maximum patch distance.

**Training Details**

- **Dataset:**
  Trained on defect-free wood images only.

- **Backbone Choices:**
  Supports ResNet18 and WideResNet50_2; in our case we used WideResNet50_2

- **Coreset Sampling:**
  Controls memory bank size with a coreset ratio(our 0.05), balancing accuracy and speed. This will take too much time while training unless chosen properly.

- **Nearest Neighbor Search:**
  Uses FAISS with L2 distance for scalable and fast search among large feature banks.

#### Performance Metrics

- **Results:**
  - F1 Score: 0.84
  - ROC AUC: 0.89
  - Optimal Threshold: 6.7

### Model 6: Variational Autoencoder (VAE)

#### Model Overview

The VAE model uses a probabilistic encoder-decoder architecture to learn a compressed latent representation of normal wood images. At test time, anomalies are detected by measuring how well the model reconstructs input images:

- **Anomalies** tend to have higher reconstruction error or deviate more in the latent distribution.

#### Architecture Details

- **Encoder:**
  - 4 Conv2D layers with strides=2, LeakyReLU activations, BatchNorm
  - Flatten + two Dense layers outputting `z_mean` and `z_log_var` (latent distribution parameters)
  - Sampling layer implements the re-parameterization trick to sample latent vector `z`.
- **Decoder:**
  - Dense layer reshaping to 16×16×64 feature map

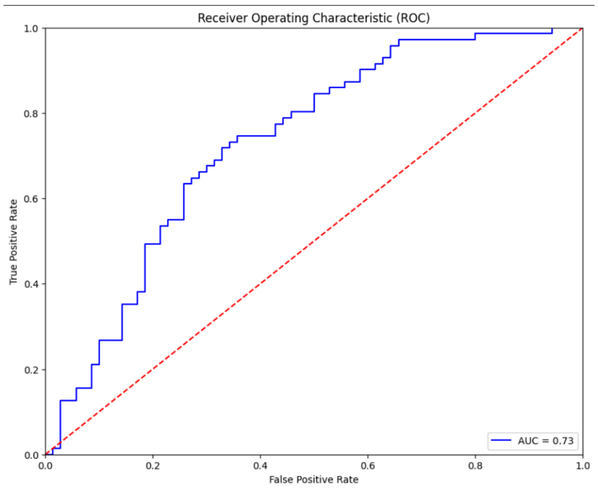
    - 4 Conv2DTranspose layers upsampling back to 256×256×1 with LeakyReLU and BatchNorm
    - Output uses sigmoid activation for pixel-wise reconstruction.
- **Loss:**
    - Reconstruction loss (binary cross-entropy scaled appropriately or MSE)
    - KL divergence between approximate posterior and prior to regularize latent space
    - Weighted sum with beta factor (set to 1.0 here).

**Training**

- Dataset: Normal wood images for training, with a validation split
- Optimizer: Adam with learning rate 1e-4
- Early stopping and learning rate reduction callbacks used
- Trained for up to 100 epochs with batch size 16

**Performance Metrics**

- **Results:**
    - F1 Score: 0.69
    - ROC AUC: 0.73
    - Optimal Threshold: 0.049



**Model Saving and Loading**

- Encoder and decoder saved separately as `.h5` files
- Results saved in `.npz` including metrics and hyperparameters
- Alternative fallback to save only weights if full model save fails