Freudenberg Use Case

Leather Defect Detection and Classification

Introduction

- Leather can be damaged during the manufacturing process
- Detect and classify thar defect to know what to do with the piece of leather
- Automate the process
- Machine Learning with different Neuronal Network
- Achieve an acceptable performance

Use Case - Problem Statement

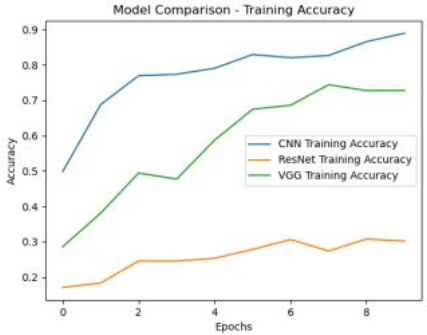
- Leather can be damaged in different ways during the manufacturing process
- Identify the defect and classify it
- Organize an assembly line to detect and sort the leather piece if we need to save it or remove it from the process.

- Leather can be damaged in different ways during the manufacturing process
- Identify the defect and classify it
- Organize an assembly line to detect and sort the leather piece if we need to save it or remove it from the process.

Machine Learning

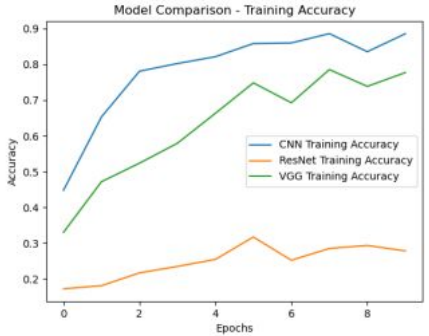
- Python with Tensorflow and Keras
- Public Dataset 3600 images, resized to 244x244 pixels
- Personal Convolutional Neuronal Network
- Resnet50
- Visual Geometry Group
- Learning function will vary between ReLu or SeLu
- On Dense layers we will always use softmax for the learning
- Batch size will vary between 32 and 64
- Always 10 Epochs

```python
def create_cnn_model(inputShape, numClasses, activationNotDense):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation=activationNotDense, input_shape=inputShape))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation=activationNotDense))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation=activationNotDense))
    model.add(layers.Flatten())
    model.add(layers.Dense(128, activation=activationNotDense))
    model.add(layers.Dense(numClasses, activation='softmax'))
    return model

def create_resnet_model(inputShape, numClasses, activationNotDense):
    base_model = tf.keras.applications.ResNet50(input_shape=inputShape, include_top=False, weights='imagenet')
    base_model.trainable = False

    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(numClasses, activation='softmax')
    ])
    return model

def create_vgg_model(inputShape, numClasses, activationNotDense):
    model = models.Sequential()
    model.add(layers.Conv2D(64, (3, 3), activation=activationNotDense, input_shape=inputShape))
    model.add(layers.Conv2D(64, (3, 3), activation=activationNotDense))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation=activationNotDense))
    model.add(layers.Conv2D(128, (3, 3), activation=activationNotDense))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(256, (3, 3), activation=activationNotDense))
    model.add(layers.Conv2D(256, (3, 3), activation=activationNotDense))
    model.add(layers.Conv2D(256, (3, 3), activation=activationNotDense))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(512, activation=activationNotDense))
    model.add(layers.Dense(numClasses, activation='softmax'))
    return model
```



(a) 32 batch ReLu

(b) 32 batch SeLu



(c) 64 batch ReLu

Results: CNN - ResNet50 - VGG

Future Work and Improvements

- Make ‘selu’ algorithm work as intended
  - Normalizing as needed our Neuronal Network
  - Using models given by the Keras library.
- Try to focus on one of the defect that we think will be more common on our manufacturing process
- Work on a Leather Classification system by texture.

- Private datasets really recent discoveries