Open in app ↗

Medium Search Write 2 A

Get unlimited access to the best of Medium for less than $1/week. Become a member

Understanding Activation Layers in Deepfake Detection Using a CNN

Dekhane Aishwarya
3 min read · Sep 29, 2024

8

In the realm of machine learning, particularly in convolutional neural networks (CNNs), activation layers play a crucial role in capturing intricate patterns within images. For a deepfake detection project, visualizing these activation layers can provide insights into how the model learns and distinguishes between authentic and manipulated images.

What Are Activation Layers?

Activation layers apply non-linear transformations to the outputs of convolutional layers, enabling the network to capture complex patterns and features. In this article, we will delve into visualizing these layers to unravel how our deepfake detection model interprets data.

Project Overview

Let's start by outlining project. This will be utilizing a CNN to classify images as either 'Original' or 'Deepfake'. Below is the code snippet for loading our dataset:

```
# Function to load images and assign labels
def load_images_from_folder(folder, label):
    images = []
    labels = []
    for filename in os.listdir(folder):
        if filename.endswith('.png'):
            img_path = os.path.join(folder, filename)
            img = Image.open(img_path).convert('RGB')
            img = img.resize((128, 128))  # Resize images for consistency
            images.append(img)
            labels.append(label)
    return images, labels

# Paths to spectrogram folders
deepfake_folder = 'filepath'
original_folder = 'filepath'

# Load images and labels
deepfake_images, deepfake_labels = load_images_from_folder(deepfake_folder, label=1)
original_images, original_labels = load_images_from_folder(original_folder, label=0)

# Combine and create dataset
images = np.array(deepfake_images + original_images)
labels = np.array(deepfake_labels + original_labels)

# Convert labels to tensor
labels = torch.tensor(labels)

# Define transformations
transform = transforms.Compose([
    transforms.ToTensor()
])
```

Defining the CNN Model

Our CNN model will comprise several convolutional layers with ReLU activation functions and max-pooling layers. Here is the model definition code:

```python
# Define the CNN model
class SpectraCNN(nn.Module):
    def __init__(self):
        super(SpectraCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 3, 1, 1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(16, 32, 3, 1, 1)
        self.conv3 = nn.Conv2d(32, 64, 3, 1, 1)
        self.fc1 = nn.Linear(64 * 16 * 16, 128)
        self.fc2 = nn.Linear(128, 2)

    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = self.pool(torch.relu(self.conv3(x)))
        x = x.view(-1, 64 * 16 * 16)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Training the Model

We will train our model with the provided training set. However, visualizations of activation layers are what will give us insights.

```python
# Training the model
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    epoch_loss = running_loss / len(train_loader)
    epoch_accuracy = 100 * correct / total
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Accuracy: {epoch_accuracy:.2f}%')
```

Visualizing Activation Layers

By activating specific layers in our network, we can visualize what the CNN focuses on when processing an image. This process helps in understanding feature extraction at each layer.

```python
# Function to register hooks for intermediate layers
def register_hooks(model, layer_names):
    activations = {}

    def get_activation(name):
        def hook(model, input, output):
            activations[name] = output.detach()
        return hook

    hooks = []
    for name in layer_names:
        layer = model._modules.get(name)
        if layer is not None:
            hooks.append(layer.register_forward_hook(get_activation(name)))
    return activations, hooks

# Get sample image from the test set
sample_image = test_images[0]
sample_image = transform(sample_image).unsqueeze(0).to(device)

# Register hooks to capture activations
activations, hooks = register_hooks(model, ['conv1', 'conv2', 'conv3'])

# Pass the sample image through the model to capture activations
model.eval()
with torch.no_grad():
    _ = model(sample_image)

# Remove hooks
for hook in hooks:
    hook.remove()
```

Insights from Visualization

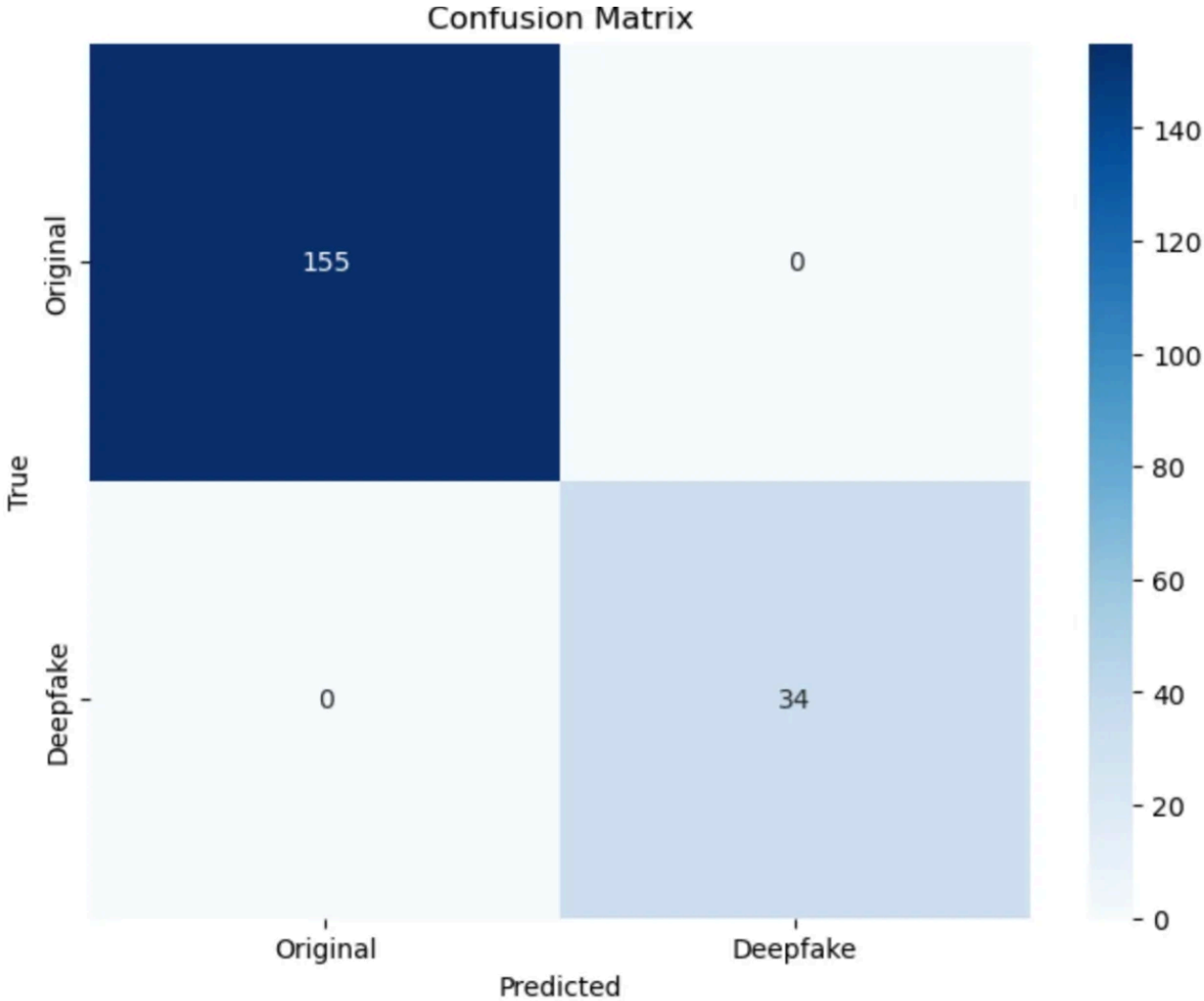
Visualizing the activations at different convolutional layers allows us to observe the following:

1. Early Layers: In the initial layers (like `conv1`), activations typically capture basic features such as edges, textures, and colors. These layers serve as the building blocks for more complex patterns.

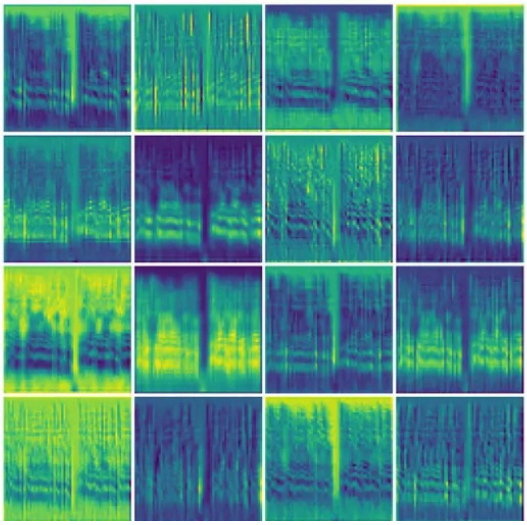
2. Middle Layers: As we move to intermediate layers (e.g., `conv2`), the features become more abstract. They start to represent combinations of the basic features, capturing patterns such as shapes, contours, and simple textures.

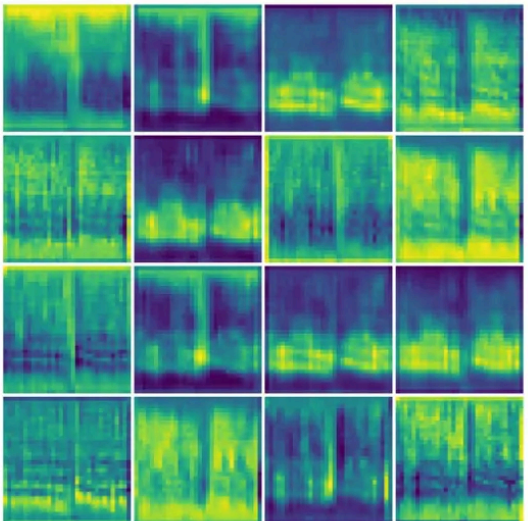
3. Deep Layers: In the deeper layers (`conv3`), the filter activations often highlight complex patterns that are highly specific to the detection task. For deepfake detection, this might involve discerning inconsistencies in textures, unnatural patterns, or artifacts introduced during the manipulation process.

These visualizations reveal how the CNN evolves its understanding from simple to complex features, ultimately allowing it to effectively distinguish between original and deepfake images.



	precision	recall	f1-score	support
Original	1.00	1.00	1.00	155
Deepfake	1.00	1.00	1.00	34
accuracy			1.00	189
macro avg	1.00	1.00	1.00	189
weighted avg	1.00	1.00	1.00	189





Conclusion

By visualizing the activation layers of a CNN used for deepfake detection, we gain valuable insights into the model's inner workings. This understanding can help in debugging, improving model performance, and building trust in the model's decisions. Activation visualization serves as a crucial tool in the arsenal of deep learning practitioners, providing a window into the model's feature extraction process.

Through this article, we explored the entire deepfake detection pipeline, from data loading and model training to the visualization of activation layers, underscoring their importance in deep learning projects.

Feel free to adapt this article with your specific dataset paths, adjust hyperparameters, and include more detailed insights tailored to your findings.

Deepfakes | Artificial Intelligence | Deep Learning

Written by Dekhane Aishwarya

1 Follower · 2 Following

Edit profile

No responses yet

What are your thoughts?

Respond

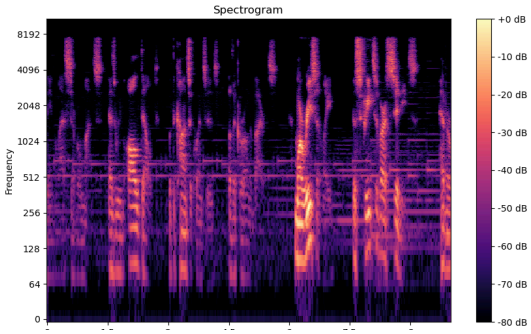
More from Dekhane Aishwarya

Dekhane Aishwarya

Utilizing Image Morphology for Enhanced Deepfake Detection in...

Deepfake detection has become a significant area of research due to the increasing...

Jul 23, 2024

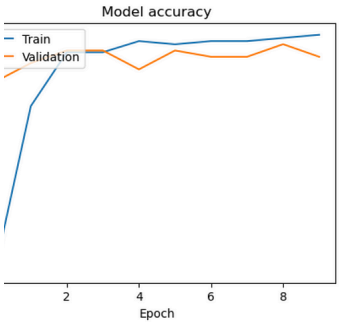


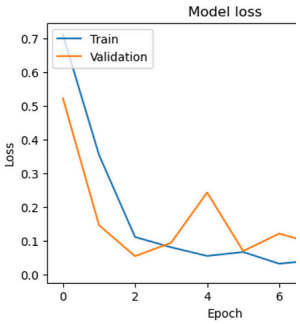
Dekhane Aishwarya

Deepfake Detection Using Spectrogram Analysis: An...

In recent years, the rise of deepfake technology has presented significant...

Jul 15, 2024





Dekhane Aishwarya

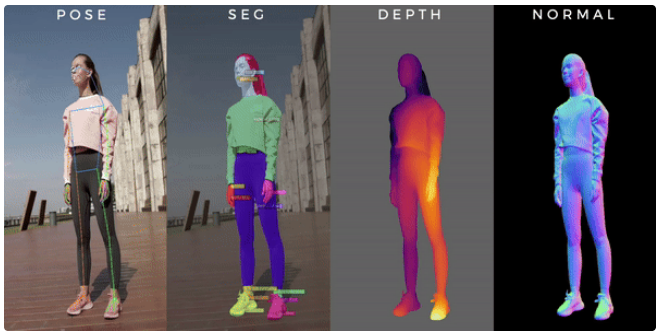
Detecting Deepfake Audio with Spectrogram Analysis Using...

Introduction

Aug 1, 2024

See all from Dekhane Aishwarya

Recommended from Medium

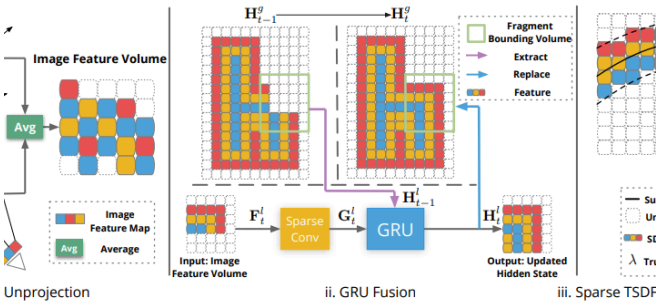


AI Papers Academy

Sapiens by Meta AI: Foundation for Human Vision Models

In this post we dive into Sapiens, a new family of computer vision models by Meta AI that...

Aug 26, 2024 130



The useless channel

Neural Recon: 4D reconstruction

NeuralRecon: The latest method for real-time 3D scene reconstruction from a monocular...

Aug 20, 2024

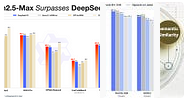
Lists

AI Regulation

6 stories · 679 saves

ChatGPT

21 stories · 957 saves



Natural Language Processing

1908 stories · 1566 saves

Generative AI Recommended Reading

52 stories · 1630 saves

In Python in Plain English by Deepak N R

YOLOv8 vs YOLOv11: A Comparison

YOLO models have been state-of-the-art in computer vision for real-time object...

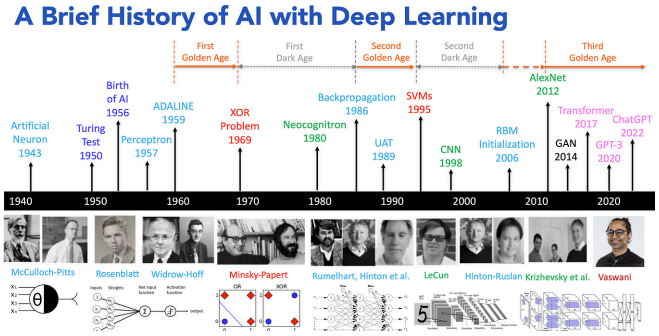
Oct 9, 2024 197 4

In The Deep Hub by Jorgecardete

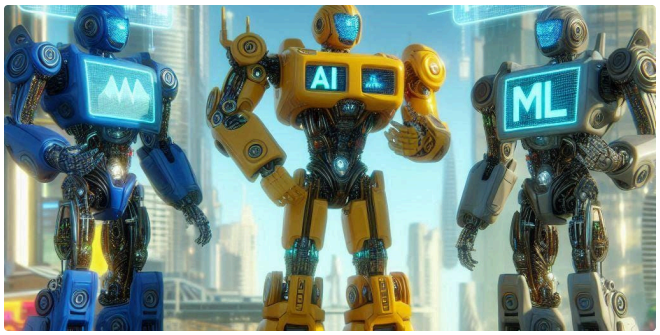
Convolutional Neural Networks: A Comprehensive Guide

Exploring the power of CNNs in image analysis

Feb 7, 2024 2.7K 38



LM Po



Brij Bhushan Singh

A Brief History of AI with Deep Learning

Artificial intelligence (AI) and deep learning have seen remarkable progress over the pas...

Sep 1, 2024 433 8

Understanding Transformer Models: A Revolution in Deep...

Transformer models have revolutionized the field of deep learning since their introductio...

Aug 3, 2024 53

See more recommendations