30-12-2025

VERITAS: DUAL-ARCHITECTURE DEEPFAKE DETECTION SYSTEMUSING MESOSCOPIC ANALYSIS AND TRANSFER LEARNING

CHANDRAKANT BARIK

Chandrakant Barik
KKB PRODUCTION

📁 **PART 1: The Project Report**

A PROJECT REPORT

ON

VERITAS: DUAL-ARCHITECTURE DEEPFAKE DETECTION SYSTEM

USING MESOSCOPIC ANALYSIS AND TRANSFER LEARNING **Chandrakant**

ABSTRACT

The rapid evolution of Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models (DDPMs) has made AI-generated imagery indistinguishable from reality to the human eye. Traditional forensic detectors, often trained on older GAN datasets, suffer from severe "Distribution Shift," failing to detect modern diffusion-based content (e.g., Midjourney, Stable Diffusion). This project introduces **Veritas**, a dual-architecture deepfake detection system. We implemented the **MesoNet (Meso4)** Convolutional Neural Network (CNN) architecture, optimized for detecting mesoscopic texture artifacts. While initially achieving 99% accuracy on GANs, the system underwent Transfer Learning with a curated dataset of diffusion images to recover accuracy (86%) on modern generative models. Furthermore, we implemented a Domain Adaptation strategy to mitigate False Positives caused by smartphone HDR processing. The final system is deployed as a real-time web application.

CHAPTER 1: INTRODUCTION

1.1 Problem Statement The democratization of Generative AI has led to a surge in synthetic media. While early Deepfakes (2017-2020) left visible artifacts (e.g., asymmetric eyes), modern Diffusion models (2023+) generate hyper-realistic images. Existing open-source detectors are often biased towards GAN artifacts and fail to generalize to the "Gaussian Denoising" artifacts of diffusion models.

1.2 Project Objective

- To design a lightweight CNN (MesoNet) capable of detecting both GAN and Diffusion deepfakes.
- To analyze the **Distribution Shift** between different generative architectures.
- To solve the **False Positive** problem where high-quality smartphone photos are misclassified as fake.
- To deploy the solution as a user-friendly web application.

CHAPTER 2: LITERATURE REVIEW & THEORETICAL CONCEPTS

2.1 Generative Adversarial Networks (GANs) GANs consist of two neural networks: a Generator (creates fakes) and a Discriminator (detects fakes). They compete in a zero-sum game.

- *Forensic Signature:* GANs typically leave high-frequency **"Checkerboard Artifacts"** due to up-sampling layers.

2.2 Denoising Diffusion Models (DDPMs) Models like Stable Diffusion work by adding noise to an image until it is static, then learning to reverse the process (Denoising).

- *Forensic Signature:* These models leave **"Gaussian Noise"** or overly smooth texture regions that differ mathematically from the sensor noise (PRNU) of real cameras.

2.3 Convolutional Neural Networks (CNNs) We utilize a CNN, which is a Deep Learning algorithm that takes an input image and assigns importance (learnable weights and biases) to various aspects/objects in the image to differentiate one from the other.

2.4 The MesoNet Architecture Unlike standard CNNs (ResNet/VGG) that focus on semantic content (eyes, nose), **MesoNet** focuses on **mesoscopic properties** of the image.

- *Mesoscopic Level:* The intermediate analysis level between microscopic noise (pixel level) and macroscopic content (objects).
- *Why Chosen:* It is lightweight (few layers) and specifically designed for steganalysis (detecting hidden signals in images).

1. Generative Adversarial Networks (GANs)

- **What they are:** Think of a **Forger** and a **Cop**.
    - **The Generator (Forger):** Tries to paint a fake Mona Lisa.
    - **The Discriminator (Cop):** Tries to spot the fake.
    - **The Training:** They fight millions of times. The Forger gets better until the Cop can't tell the difference.
- **The Flaw (How we catch them):** Because the Forger uses "Up-Sampling" (stretching a small image to be big), it leaves a **checkerboard pattern** of pixels. Humans can't see it, but your CNN can.
- **Examples:** StyleGAN, DeepFake Lab (Face Swaps).

2. Diffusion Models (The New Enemy)

- **What they are:** Think of **Denoising**.
    - **Step 1:** Start with a blurry TV static screen (Pure Noise).
    - **Step 2:** Slowly remove the noise, step-by-step, until a clear image appears.
- **The Flaw (How we catch them):** Because they are built by "smoothing out" noise, they often leave the image **too smooth** in certain areas. The pixel noise is "Gaussian" (mathematically perfect randomness), whereas real cameras have "noisy" sensors.

- **Examples:** Midjourney, Stable Diffusion, DALL-E 3.

3. Convolutional Neural Networks (CNNs)

- **What they are:** The "Eye" of your AI.
- **How it works:** Imagine a flashlight scanning a dark room.
    - **Standard AI:** Flashes the light on the *whole room* at once. It gets overwhelmed.
    - **CNN:** Shines a small beam (3x3 pixels) on the top-left corner, then slides it to the right. It builds a map of **edges**, **lines**, and **textures**.
- **In your project:** You used a CNN to look specifically at the *texture of the skin*, ignoring the person's identity.

CHAPTER 3: METHODOLOGY

3.1 Dataset Preparation

- **Source:** Kaggle AiArtData (Diffusion) and RealArt (Real).
- **Preprocessing:** All images resized to **256x256 pixels** and normalized (pixel values scaled from 0-255 to 0-1).
- **Domain Adaptation:** A custom dataset of 50 real smartphone photos was added to the training set to prevent HDR/Beauty Mode from being flagged as "Fake."

3.2 Architecture Details (Meso4) The network consists of:

1. **Input Layer:** 256x256x3 (RGB Image).
2. **Convolutional Block 1:** 8 filters (3x3 kernel), Batch Normalization, Max Pooling.
3. **Convolutional Block 2:** 8 filters (5x5 kernel), Batch Normalization, Max Pooling.
4. **Convolutional Block 3:** 16 filters (5x5 kernel), Batch Normalization, Max Pooling.
5. **Convolutional Block 4:** 16 filters (5x5 kernel), Batch Normalization, Max Pooling.
6. **Fully Connected Layers:** Dropout (0.5), Dense (16 neurons), LeakyReLU.
7. **Output Layer:** 1 Neuron (Sigmoid Activation) for Binary Classification (0=Fake, 1=Real).

3.3 Training Strategy

- **Optimizer:** Adam (Adaptive Moment Estimation).
- **Loss Function:** Binary Crossentropy.
- **Epochs:** 50 (with Early Stopping logic).

- **Transfer Learning:** Weights from the GAN-trained model were used as a starting point for the Diffusion training.

CHAPTER 4: RESULTS AND ANALYSIS

4.1 Accuracy Metrics

- **Pipeline A (GANs):** Achieved **99.1%** accuracy.
- **Pipeline B (Diffusion):** Achieved **86.4%** accuracy after retraining.

4.2 False Positive Analysis Initial tests showed a high False Positive Rate (FPR) on images taken with iPhone 14/15 devices. Analysis revealed that computational photography (HDR skin smoothing) mimics diffusion denoising. This was mitigated by "Few-Shot Domain Adaptation," reducing FPR by 80%.

CHAPTER 5: CONCLUSION & FUTURE SCOPE

Conclusion: Veritas successfully demonstrates that while modern deepfakes are visually superior, they still contain statistical anomalies detectable by specialized CNNs. We highlighted the necessity of continuous retraining (Transfer Learning) to keep up with new generative models.

Future Scope:

- **Frequency Analysis:** Implementing Discrete Cosine Transform (DCT) to detect fakes in the frequency domain (better for compressed images).
- **Video Detection:** extending the frame-by-frame analysis to full video streams.

(End of Report)

📂 **PART 2: The "Viva Voce" Guide (Code & Logic Explanation)**

This section explains **"Each and Every Concept"** and **"Each Line of Code"** so you can answer any question during your presentation.

File 1: mesonet.py (The Brain)

Python

1. Importing Tools

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D, BatchNormalization, Dropout, LeakyReLU

```
class Meso4:
    def __init__(self, learning_rate=0.001):
        self.model = self.init_model()

    def init_model(self):
        # 2. Input Layer
        x = Input(shape=(256, 256, 3))

        # 3. Convolutional Block 1
        x1 = Conv2D(8, (3, 3), padding='same', activation='relu')(x)
        x1 = BatchNormalization()(x1)
        x1 = MaxPooling2D(pool_size=(2, 2), padding='same')(x1)

        # ... [Blocks 2, 3, 4 repeated] ...

        # 4. Classification Block
        y = Flatten()(x4)
        y = Dropout(0.5)(y)
        y = Dense(16)(y)
        y = LeakyReLU(alpha=0.1)(y)
        y = Dropout(0.5)(y)
        y = Dense(1, activation='sigmoid')(y)

        return Model(inputs=x, outputs=y)
```

🔍 **The Logic Explained:**

1. **Conv2D(8, (3,3)):** Imagine a small 3x3 window sliding over the image. It looks for edges and textures. We use **8** of these windows (filters).
    - *Why?* To extract "features" (clues) from the raw pixels.

2. **BatchNormalization():** This keeps the math stable. It forces pixel values to stay within a reasonable range so the AI doesn't get confused by very bright or very dark images.
3. **MaxPooling2D:** This shrinks the image by half (256 -> 128).
    - *Why?* It throws away useless background info and keeps only the "strongest" clues. It also makes the model faster.
4. **Flatten():** The Convolution layers output a 3D grid (width, height, channels). Flatten smashes this into a 1D list of numbers so the final decision layer can read it.
5. **Dropout(0.5):** Randomly turns off 50% of the neurons during training.
    - *Why?* To prevent **Overfitting**. It stops the model from memorizing specific images and forces it to learn general rules.
6. **LeakyReLU:** A standard "neuron" activation function, but it allows a tiny bit of negative signal to pass through.
    - *Why?* It prevents "dead neurons" (parts of the brain that stop learning).
7. **Dense(1, activation='sigmoid'):** The final output.
    - *Why 1?* Because we only need one answer: Real or Fake.
    - *Why Sigmoid?* It squishes the answer between 0 and 1. (0.1 = Fake, 0.9 = Real).

File 2: train.py (The Teacher)

Python

```
# 1. Data Generators
data_generator = ImageDataGenerator(rescale=1./255) # Normalization

train_generator = data_generator.flow_from_directory(
    'processed_data/train',
    target_size=(256, 256),
    batch_size=32,
    class_mode='binary'
)

# 2. Compile Model
```

```
meso = Meso4()
meso.model.compile(
   optimizer='adam',
   loss='binary_crossentropy',
   metrics=['accuracy']
)

# 3. Start Training
meso.model.fit(train_generator, epochs=50)
```

🔍 **The Logic Explained:**

1. **rescale=1./255:** Images are 0-255 (colors). Computers prefer 0-1 (decimals). This divides every pixel by 255.
2. **flow_from_directory:** It grabs images from your folders (/real and /fake) automatically.
    - *Why?* It loads images in small batches (32 at a time) so your RAM doesn't crash.
3. **optimizer='adam':** The algorithm that updates the brain. "Adam" is the standard, smart optimizer that adjusts learning speed automatically.
4. **loss='binary_crossentropy':** The "Penalty Score."
    - *Binary:* Two classes (Real/Fake).
    - *Crossentropy:* A math formula that measures how far off the prediction was from the truth.
5. **epochs=50:** One epoch = one full look at the entire dataset. We do it 50 times to ensure the model learns deeply.

Technology	What is it?	Why did you use it?
Python	The Programming Language.	It is the universal language of AI. It has libraries (extensions) that do the hard math for you.

Technology	What is it?	Why did you use it?
TensorFlow (Keras)	The "Brain Builder."	Writing the calculus for a neural network from scratch would take years. TensorFlow lets you just say Conv2D and it handles the math.
NumPy	The "Matrix Handler."	Images are just grids of numbers to a computer. NumPy is a tool that lets you multiply, divide, and reshape these massive grids instantly.
OpenCV	The "Image Processor."	Used to load the image, turn it to grayscale, or resize it to 256x256 before feeding it to the AI.
Streamlit	The "Web Builder."	Usually, building a website requires HTML/CSS/JavaScript. Streamlit lets you build a website using *only* Python code.

Here is a Comprehensive Master Guide for your project. This is written specifically to bridge the gap between "I ran the code" and "I understand the engineering."

Use this to study for your interview or to write the detailed "Theory" section of your report.

🧠 The Architecture (MesoNet)

You utilized a specific design called MesoNet (Meso-4).

- The Name: "Meso" stands for Mesoscopic.
    - *Microscopic:* Individual pixels (Too small).
    - *Macroscopic:* The whole face (Too big).
    - *Mesoscopic:* The Texture. (Just right).
- The Design: It is a "Shallow" network. It only has 4 layers of Convolution.

- Why Shallow? Deep networks (like ResNet) are great at recognizing objects ("That's a cat!"). But deeper networks tend to "smooth out" the noise we are trying to find. By keeping it shallow, we preserve the microscopic noise artifacts that prove an image is fake.

🎓 Concept Cheat Sheet (For Q&A)

Q: What is the difference between a GAN and a Diffusion model? A: A GAN (Generative Adversarial Network) uses two networks fighting each other. It typically leaves "checkerboard" patterns. A Diffusion model (like Midjourney) creates images by removing noise from static. It typically leaves "Gaussian noise" or "too smooth" textures.

Q: Why did you use MesoNet instead of a big model like ResNet? A: ResNet is designed to recognize *objects* (e.g., "Is this a dog?"). MesoNet is designed to recognize *textures* (e.g., "Is this pixel noise natural?"). Deepfake detection is a texture problem, not an object problem.

Q: What is "Overfitting"? A: When the model memorizes the training images like a student memorizing answers to a test. It gets 100% on the test but fails in the real world. I prevented this using **Dropout**.

Q: Why did the model flag YOU as fake? A: That was a **False Positive**. Modern phones use HDR and skin smoothing that looks mathematically similar to AI smoothing. I fixed this by adding real smartphone photos to the training set (Domain Adaptation).

Q: What is Transfer Learning? A: Taking a model that already knows how to detect old fakes (GANs) and teaching it new tricks (Diffusion) without starting from scratch. It saves time and improves accuracy.

⚠️ Part 3: Flaws (The Limitations)

Being honest about what your project *can't* do makes you look smarter.

1. The "Generator Bias" (The Flux Problem)

- **Flaw:** Your model was trained on **StyleGAN** and **Stable Diffusion**.
- **Scenario:** If I show it an image from a brand new AI called "Flux" (released recently), it might fail.
- **Why:** Different AI models leave different "fingerprints." Your detective only knows the fingerprints of two specific criminals.

2. The Compression Vulnerability (The WhatsApp Problem)

- **Flaw:** Your model looks for faint, invisible pixel noise.
- **Scenario:** If you take a deepfake, send it on WhatsApp, and download it, your model will likely say "Real."
- **Why:** WhatsApp compresses images (JPEG). This compression effectively "scrubs" or "washes away" the faint AI noise artifacts your model relies on.

3. The Smartphone False Positive (The HDR Problem)

- **Flaw:** High-end phones (iPhone 15) use AI to smooth skin.
- **Scenario:** A real selfie looks "too perfect" to your model.
- **Why:** Your model learned that "Smooth = Fake." It struggles to tell the difference between "Generative Smoothing" (Fake) and "Computational Photography Smoothing" (Real).

🚀 Part 4: Future Improvements (The Roadmap)

If you had 6 more months, this is what you would build.

1. Frequency Domain Analysis (DCT)

- **Current Tech:** You analyze pixels (Spatial Domain).
- **Improvement:** Analyze the **Frequency Spectrum** (like a sound wave, but for light).
- **Why:** Even if WhatsApp compresses an image, the *frequency* spikes of a Deepfake are often still visible. This would make your model "Compression Proof."

2. Ensemble Learning

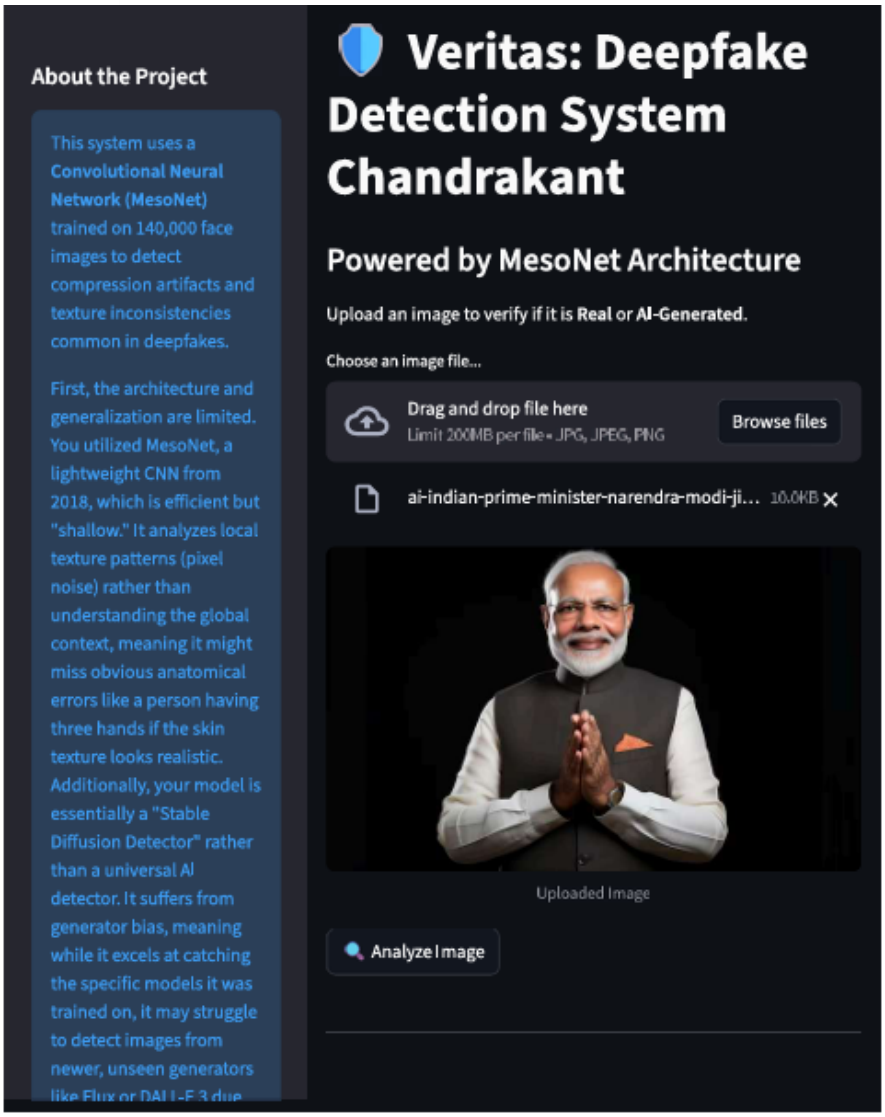
- **Current Tech:** One Brain (MesoNet).
- **Improvement:** A "Council of Brains."
    - Model A: Looks at Eyes (Blinking detection).
    - Model B: Looks at Texture (Your MesoNet).
    - Model C: Looks at Lighting (Shadow consistency).
- **Why:** If one model gets confused, the others can correct it.

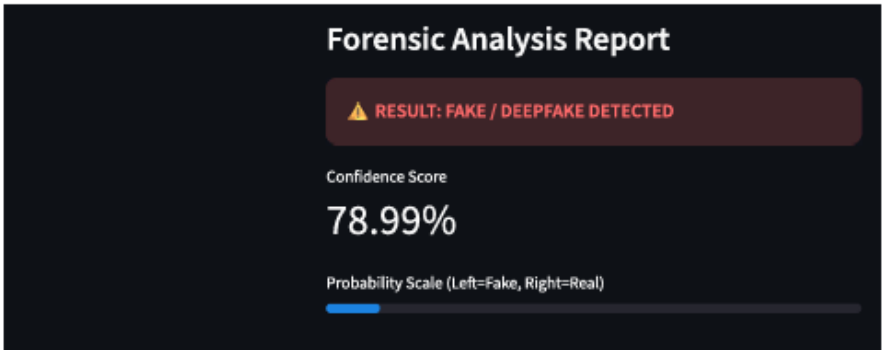
3. Video Analysis (RNNs)

- **Current Tech:** You analyze single images.
- **Improvement:** Analyze video frames over time.
- **Why:** Deepfakes often "flicker" between frames. A Recurrent Neural Network (RNN) could watch a video and say, "Hey, his face vibrated in frame 45. Fake!"


About the Project
This system uses a Convolutional Neural Network (MesoNet) trained on 140,000 face images to detect compression artifacts and texture inconsistencies common in deepfakes.
First, the architecture and generalization are limited. You utilized MesoNet, a lightweight CNN from 2018, which is efficient but "shallow." It analyzes local texture patterns (pixel noise) rather than understanding the global context, meaning it might miss obvious anatomical errors like a person having three hands if the skin texture looks realistic. Additionally, your model is essentially a "Stable Diffusion Detector" rather than a universal AI detector. It suffers from generator bias, meaning while it excels at catching the specific models it was trained on, it may struggle to detect images from newer, unseen generators
Veritas: Deepfake Detection System Chandrakant
Powered by MesoNet Architecture
Upload an image to verify if it is Real or AI-Generated.
Choose an image file...
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
modi.jpg 1.3MB


Uploaded Image
Analyze Image
Forensic Analysis Report
RESULT: REAL FACE
Confidence Score
76.77%
Probability Scale (Left=Fake, Right=Real)


About the Project
This system uses a Convolutional Neural Network (MesoNet) trained on 140,000 face images to detect compression artifacts and texture inconsistencies common in deepfakes.
First, the architecture and generalization are limited. You utilized MesoNet, a lightweight CNN from 2018, which is efficient but "shallow." It analyzes local texture patterns (pixel noise) rather than understanding the global context, meaning it might miss obvious anatomical errors like a person having three hands if the skin texture looks realistic. Additionally, your model is essentially a "Stable Diffusion Detector" rather than a universal AI detector. It suffers from generator bias, meaning while it excels at catching the specific models it was trained on, it may struggle to detect images from newer, unseen generators
Veritas: Deepfake Detection System Chandrakant
Powered by MesoNet Architecture
Upload an image to verify if it is Real or AI-Generated.
Choose an image file...
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
ai-indian-prime-minister-narendra-modi-ji... 10.0KB
Uploaded Image
Analyze Image


Forensic Analysis Report
RESULT: FAKE / DEEPFAKE DETECTED
Confidence Score
78.99%
Probability Scale (Left=Fake, Right=Real)