RETHINKING DEEPFAKE DETECTION

Gabriel Simiyu and Jazper Toralba

WE CAN NO LONGER TRUST WHAT WE SEE

- Images from GANs and Diffusion models are now virtually Indistinguishable from authentic images
- Misuse threatens political and economic domains

HOW DO GENERATIVE MODELS WORK?

GANs (Generative Adversarial Networks):

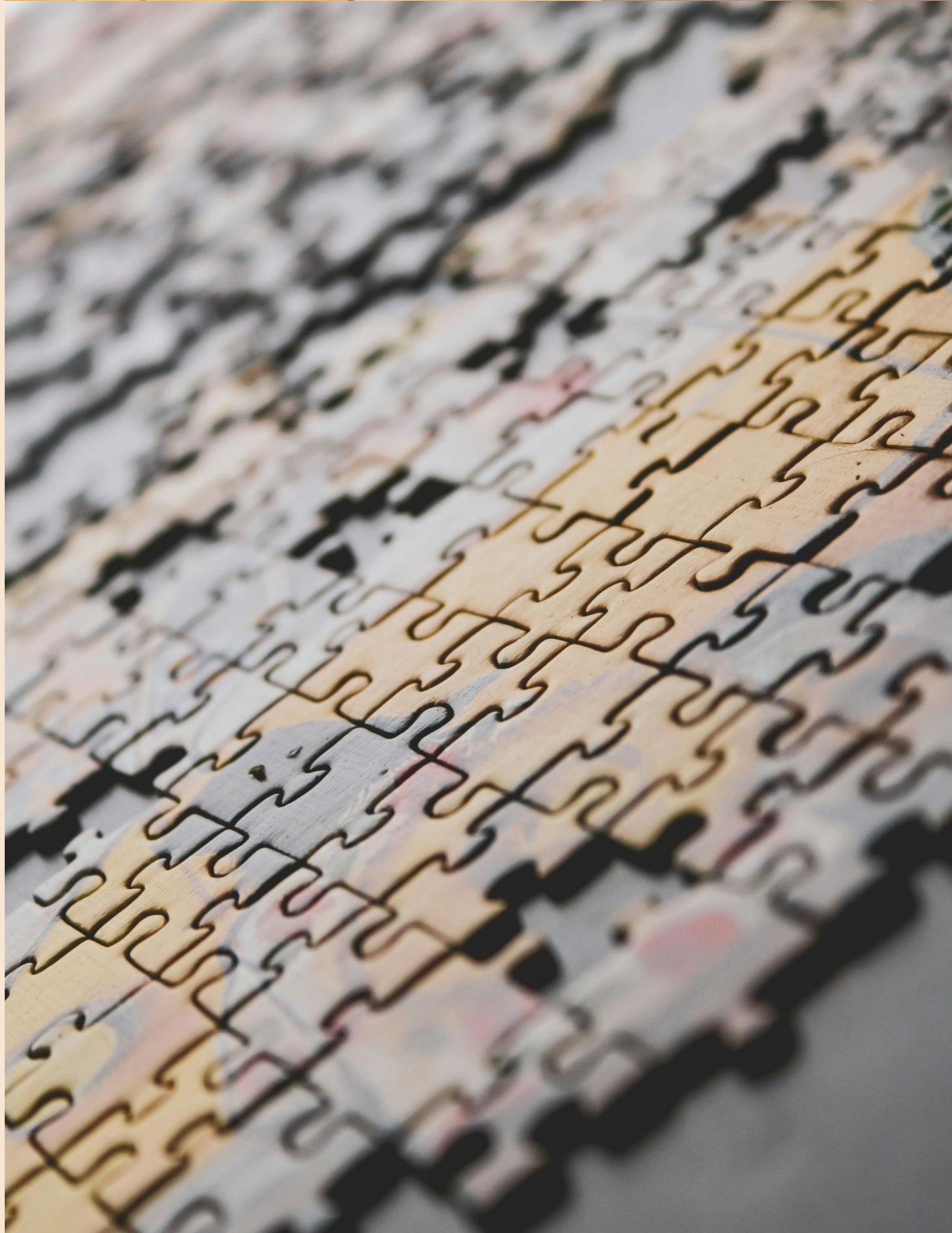
- Use up-sampling operations (nearest neighbor, bilinear) to generate high-resolution images
- Common architectures: ProGAN, StyleGAN, BigGAN, CycleGAN

Diffusion Models:

- Generate images through iterative denoising process
- Examples: Stable Diffusion, DALL-E 2, Midjourney, ADM

THE DEEPFAKE PROBLEM

- Current detectors work well on training data, **but fail catastrophically on unseen generators**
- The real world has a plethora of generation models; we must be able to generalize to keep up



THE GENERALIZATION CRISIS

THE UP SAMPLING PROBLEM

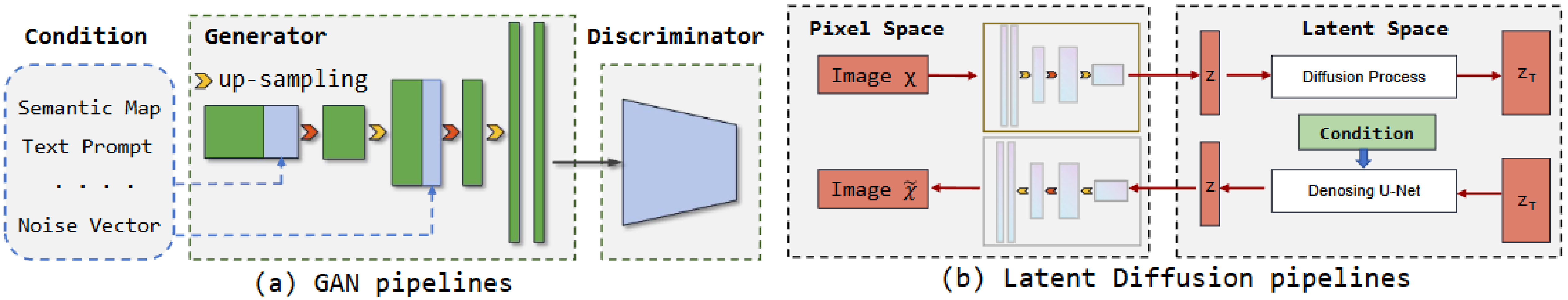
Frequency based detection (Frank, Durall, F3-Net)

- Analyzed global patterns across entire image
- Looked at spectral distribution

Image-based detection (CNN Detection, Patchfor)

- Trained on pixels directly
- Found model specific "fingerprints"

"Frequency-based artifacts are insufficient for achieving generalization detection" – FreGAN findings



LIMITATIONS

Cross-Generator Performance Breakdown:

- ProGAN -> ProGAN: 95% accuracy
- ProGAN -> StyleGAN: 63% accuracy
- ProGAN -> Diffusion: 40% accuracy

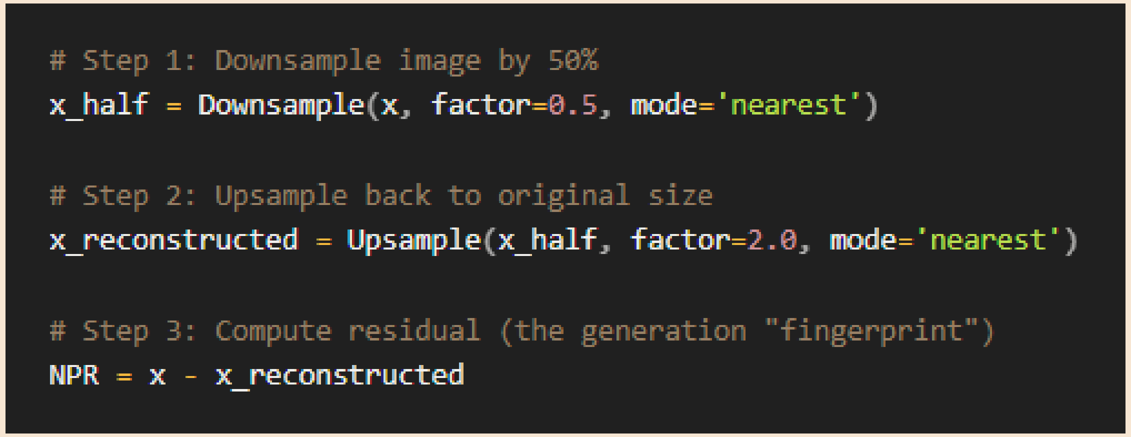
Root Cause: Overfitting to training distribution

- Frequency methods fail due to diverse patterns
- Spatial methods lack architectural understanding

We need <u>universal</u> detection even for Diffusion

NPR METHOD: A UNIVERSAL SOLUTION

CORE INNOVATION



```
# Step 1: Downsample image by 50%
x_half = Downsample(x, factor=0.5, mode='nearest')

# Step 2: Upsample back to original size
x_reconstructed = Upsample(x_half, factor=2.0, mode='nearest')

# Step 3: Compute residual (the generation "fingerprint")
NPR = x - x_reconstructed
```

- **Up-sampling is generally used everywhere**
  - All GANs use it (ProGAN, StyleGAN, BigGAN, etc.)
  - All Diffusion models use it via U-Net Decoder
- **Up-sampling transforms low resolution images to high resolution**
  - <u>This creates artifacts as LOCAL correlations between neighboring pixels</u>

Nearest neighbor interpolation makes 2×2 pixels share the same value"

NEIGHBORING PIXEL RELATIONSHIPS

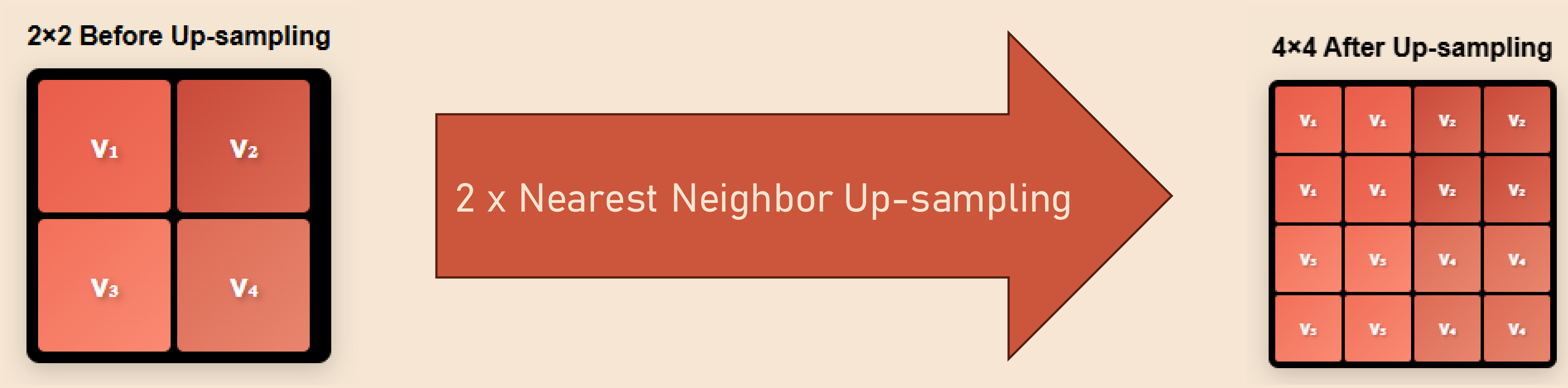
Mathematical Foundation:

- Divides images into 2x2 grids and calculates differences between neighboring pixels
- Each grid: V = [w1, w2, w3, w4]
- NPR(V) = [0, w2-w1, w3-w1, w4-w1]

Why It Detects Fakes:

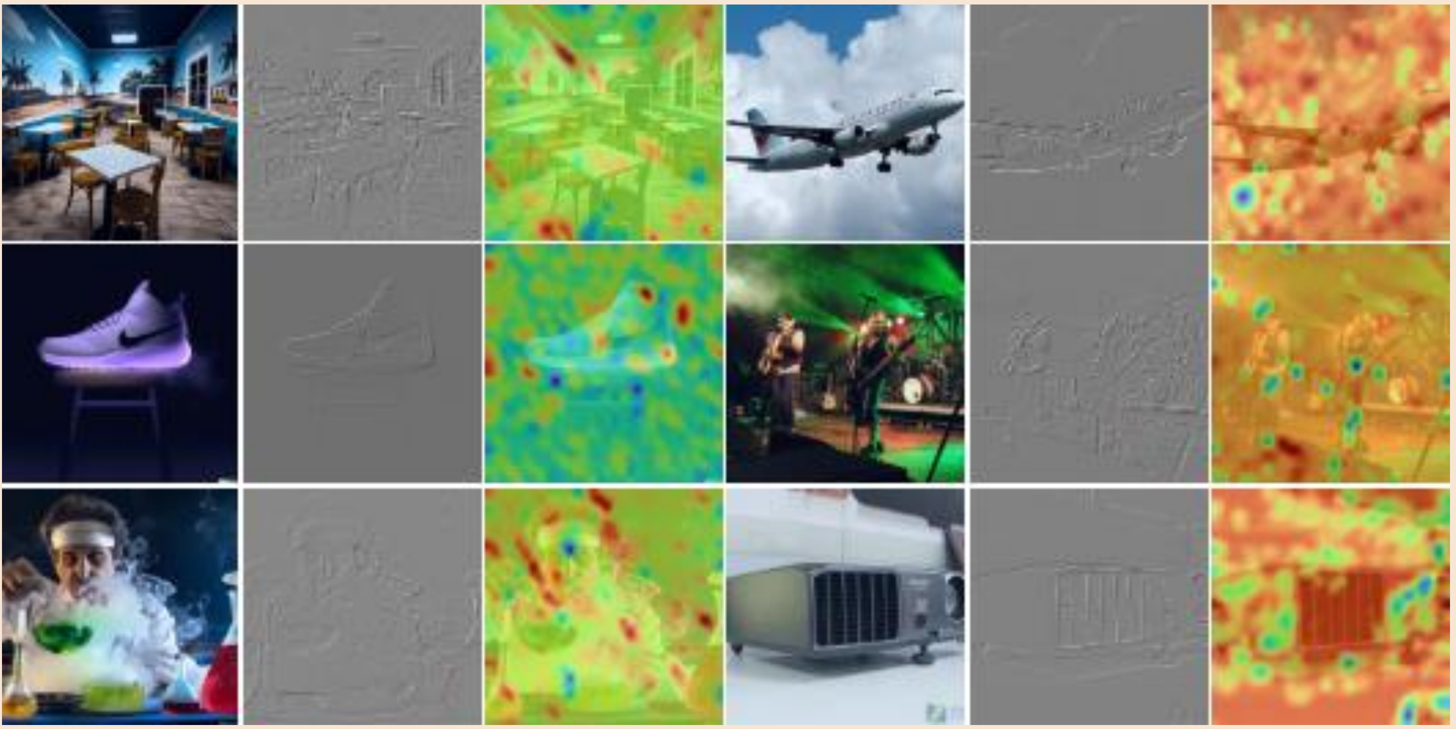
- Real photo pixels are independent
- Up sampling creates dependencies in generated images
- CNN translation invariance preserves artifacts

UP-SAMPLING OPERATION



Why this matters for NPR

- NPR Formula: V_c = {w_1 – w_j, w2 –w_j, w_3 – w_j, w_4 – w_j}
- For perfect up sampling all differences would be 0
- After CNN processing small but detectable correlation remains
- Real Images: Large NPR differences (i.e. natural variation)
- Fake Images: Small NPR differences (i.e. up-sampling residue)



WHY THIS WORKS

Three properties make NPR powerful

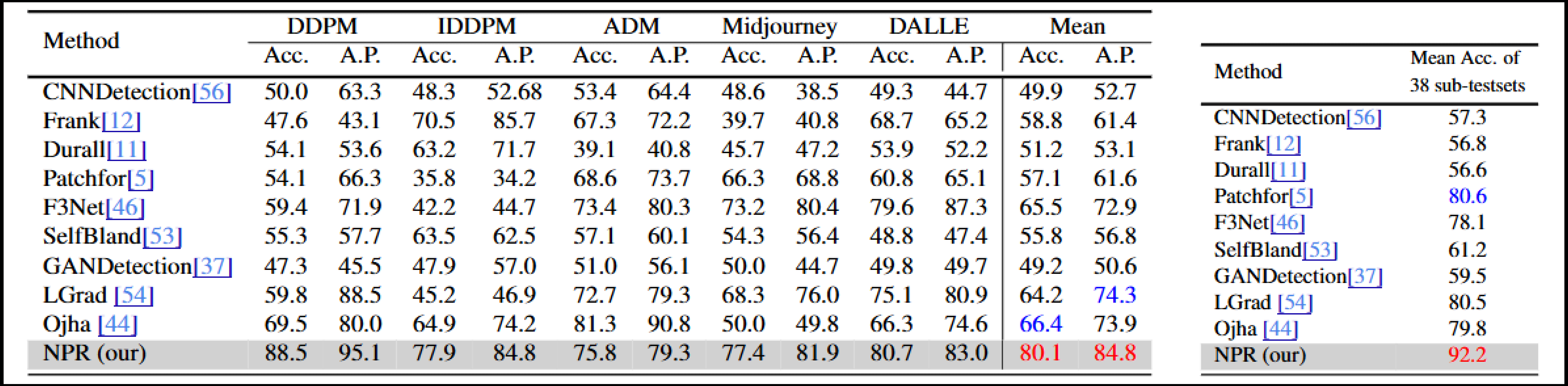
1. <u>Translation invariance</u>: CNNs preserve the up-sampling trace
2. <u>Local correlations</u>: correlation remains even after generators add detail
3. <u>Relative Measurements</u>: resilient to different image content

Experiment Design

- <u>Training</u>: Only ProGAN images
  - (4 classes: car, cat, chair, horse)
- <u>Testing</u>: 28 different Models
  - 17 GANs (including StyleGAN, BigGAN, CycleGAN, etc.)
  - 11 Diffusion models (DDPM, Stable Diffusion, DALL-E, Midjourney, etc.)

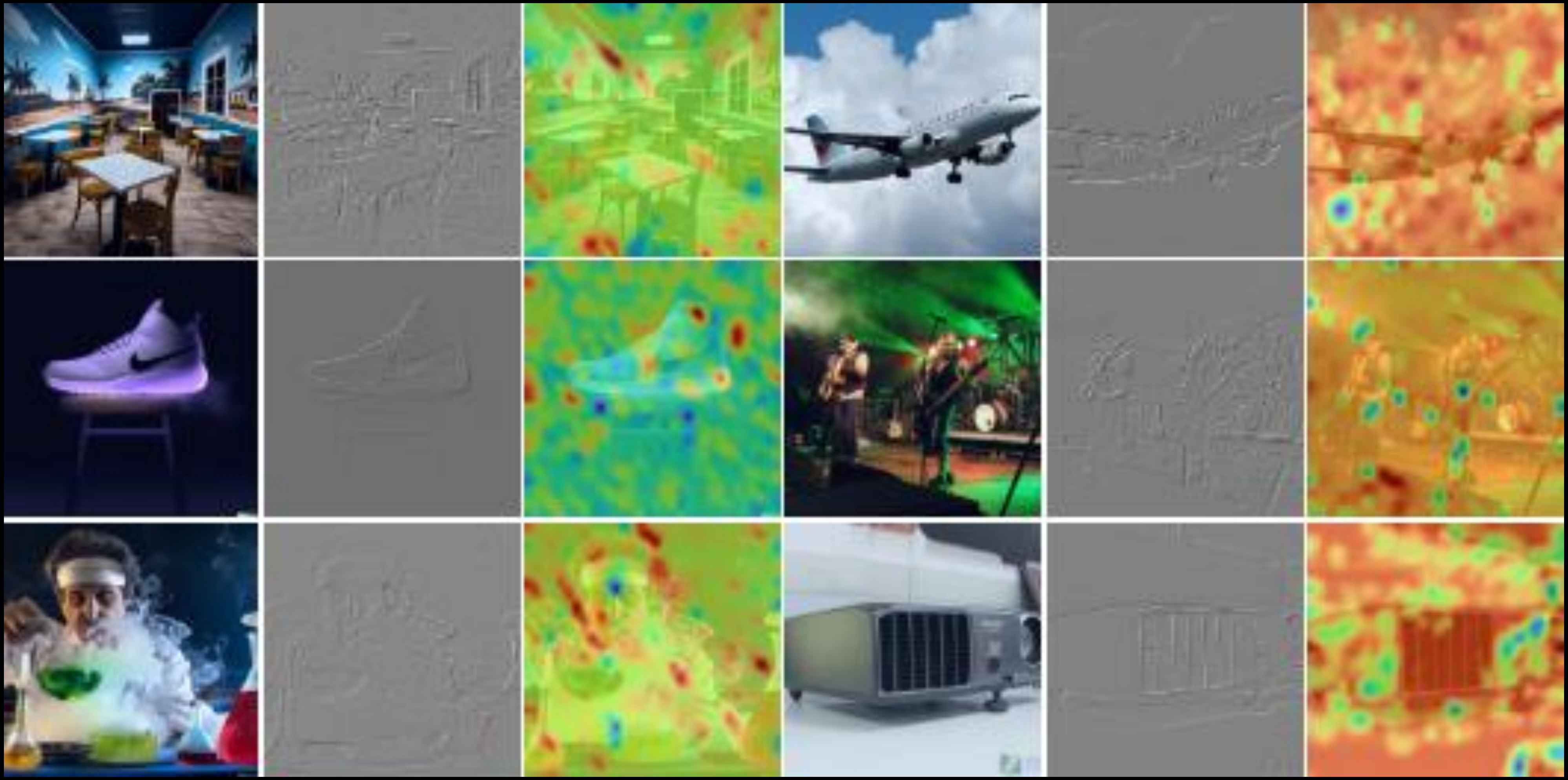
92.2%

Mean accuracy across 28 different generators using NPR with ProGAN training only



Method	DDPM		IDDPM		ADM		Midjourney		DALLE		Mean	
	Acc.	A.P.	Acc.	A.P.	Acc.	A.P.	Acc.	A.P.	Acc.	A.P.	Acc.	A.P.
CNNDetection[56]	50.0	63.3	48.3	52.68	53.4	64.4	48.6	38.5	49.3	44.7	49.9	52.7
Frank[12]	47.6	43.1	70.5	85.7	67.3	72.2	39.7	40.8	68.7	65.2	58.8	61.4
Durall[11]	54.1	53.6	63.2	71.7	39.1	40.8	45.7	47.2	53.9	52.2	51.2	53.1
Patchfor[5]	54.1	66.3	35.8	34.2	68.6	73.7	66.3	68.8	60.8	65.1	57.1	61.6
F3Net[46]	59.4	71.9	42.2	44.7	73.4	80.3	73.2	80.4	79.6	87.3	65.5	72.9
SelfBland[53]	55.3	57.7	63.5	62.5	57.1	60.1	54.3	56.4	48.8	47.4	55.8	56.8
GANDetection[37]	47.3	45.5	47.9	57.0	51.0	56.1	50.0	44.7	49.8	49.7	49.2	50.6
LGrad [54]	59.8	88.5	45.2	46.9	72.7	79.3	68.3	76.0	75.1	80.9	64.2	74.3
Ojha [44]	69.5	80.0	64.9	74.2	81.3	90.8	50.0	49.8	66.3	74.6	66.4	73.9
NPR (our)	88.5	95.1	77.9	84.8	75.8	79.3	77.4	81.9	80.7	83.0	80.1	84.8

Method	Mean Acc. of 38 sub-testsets
CNNDetection[56]	57.3
Frank[12]	56.8
Durall[11]	56.6
Patchfor[5]	80.6
F3Net[46]	78.1
SelfBland[53]	61.2
GANDetection[37]	59.5
LGrad [54]	80.5
Ojha [44]	79.8
NPR (our)	92.2



Class Activation Map (CAM) Visualization

*"The CAMs for real images highlight a **broader portion** of the image, whereas the CAMs for fake images tend to **emphasize localized regions**."*

RESULTS

Diffusion

- **Mean Accuracy:** 95.3% on Diffusion Forensics
- **Mean Accuracy:** 95.2% on Ohja Diffusion Set
- **Mean Accuracy:** 80.1% even on 1,000 step diffusions (DALLE, Midjourney)
- **Beats previous methods by 6.4%**

Trained only on ProGAN and detected Diffusion models, even being out Ojha by 20.9%

GAN

- **Mean Accuracy:** 92.5% on ForenSynths (8 GANs)
- **Mean Accuracy:** 93.2% on Self-Synthesis (9 GANs)

Beats previous methods by 6.4%, even beating methods that were trained on the same test models

THE PROBLEM

NPR Limitations

1. **Fixed Scale:** Uses 0.5 uniformly matching 2x up-sampling
2. **Suboptimal for New Models:** 2025 generators use varied up-sampling strategies
3. **No adaptive Mechanism:** Cannot adjust to input-specific characteristics

Our Hypotheses

1. *The interpolation factor (currently 0.5) affects the detection performance differently for GAN-generated vs. Diffusion-generated images. An attention mechanism can learn to automatically weight NPR scales based on input characteristics, improving detection accuracy and confidence*
2. *Attention-weighted multi-scale NPR will generalize better to unseen 2025 generators than single-scale NPR*

OUR EXTENSION

Questions

1. Does a smaller factor capture different artifacts than a larger factor?
2. Are GAN artifacts more visible at certain scales versus diffusion artifacts
3. Can we improve Generalization by using multiple scales

Relevance

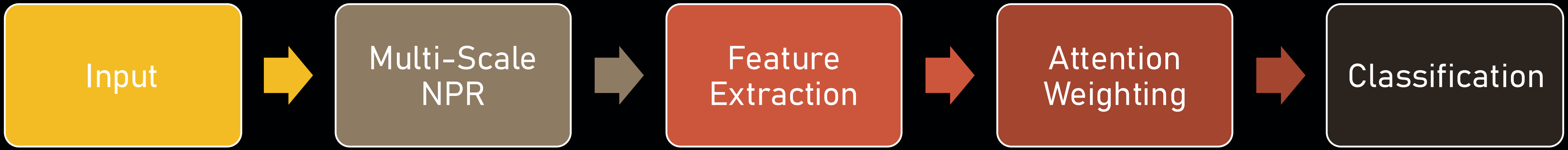
- Different generators may leave artifacts at different frequency scales
- Optimal scale for ProGAN may not be optimal for Stable Diffusion
- Multi-scale approach could capture complementary information

PROPOSED SOLUTION

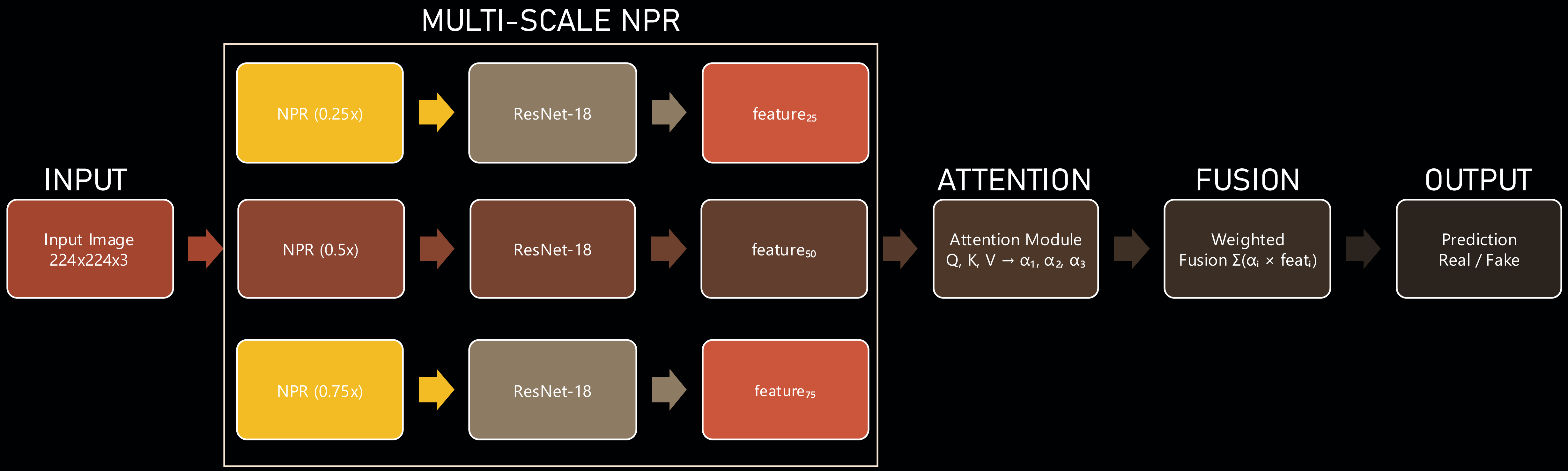
Attention-Weighted Multi-Scale NPR

Our approach introduces three key innovations

1. Multi-scale NPR extraction [0.25, 0.5, 0.75]
2. Attention mechanism for adaptive scale weighting
3. Interpretable weight visualization



MULTI-SCALE NPR ARCHITECTURE WITH ATTENTION WEIGHTING



INPUT SPECIFICATIONS

Parameter	Value	Justification
Image Size	224x224x3	Standard ImageNet dimension for ResNet-18
Batch Size	32	Optimal for 24GB GPU memory
NPR Scales	[0.25, 0.50, 0.75]	Captures low, mid, high frequency artifacts
interpolation	Bilinear	Smooth upsampling, preserves artifacts

DATA SOURCES

Training Data (140k images)

- **ForenSynths Dataset**:
  - 4 GAN classes (ProGAN, StyleGAN, StyleGAN2, ProjectedGAN)
  - 35K images per class
  - Includes aligned real images

Validation Data (40k images)

- 20% held-out from ForenSynths
- Maintains class balance

Dataset	Size	Purpose
ForenSynths Test	80K	Baseline comparison
Diffusion Forensics	60K	Cross-family testing
FLUX	500	2025 model evaluation
Midjourney v6	500	2025 model evaluation
DALL-E 3	500	2025 model evaluation

TEST DATASETS FOR HYPOTHESIS 1: 2025 NEWEST MODELS

Category	Model 1: FLUX	Model 2: Midjourney v6	Model 3: DALL-E
Release	2024 (open-source)	2024 update	2023 → integrated updates 2024
Architecture	Flow-matching (not diffusion)	Proprietary, text-to-image	Proprietary API-based
Real Images	COCO/ImageNet equivalents	Public Discord communities (ethically sourced)	Matched set from curated sources
Why?	Different architectural paradigm—true architectural diversity test	Most popular commercial model, represents real-world threat	Leading model, integrated with ChatGPT ecosystem

Expected Performance:
- Baseline (ProGAN, seen): ~95-99%
- New models (unseen): 70-85% (expected)

IMPLEMENTATION

Development

- PyTorch implementation
- Leverage homework components to speed up development

Details from Homework

- Attention mechanism from Homework 1
- GAN discriminator from Homework 3 for binary classification
- Diffusion UNET from Homework 4 to implement Scale Hypothesis

Dataset	Size	Generators	Purpose
ForenSynths Test	80K	8 GANs	Baseline comparison
Diffusion Forensics	60K	8 Diffusion	Cross-Family testing
2025 Models	1.5k	FLUX, Midjourney v6, DALL-E 3	New Architecture Testing

HARDWARE

Hardware Requirements

- **GPU:** NVIDIA RTX 2080 Super with 8GB RAM
- **RAM:** 32GB system memory
- **Storage:** 1TB for datasets

Training Time Estimates

- **Feature extraction:** Pretrained so 0 hours
- **Attention module:** 8-12 hours
- **Full pipeline:** 16-20 hours

Computational Efficiency

- **Parameters:** 1.2 million (attention) + 11 million (ResNet-18)
- **Memory:** 8GB during training

IMPLEMENTATION TIMELINE

Nov 18-24th

- Multi-Scale NPR implmentation
  - Feature extraction pipeline
  - Simple weighted fusion baseline

Nov 25th - Dec 1st

- Attention module implementation
- Training
- Initial comparison experiments

Dec 2nd - 8th

- Test on 2025 models
- Write Report and experiment findings

DELIVERABLES DECEMBER 9TH

Code Deliverables

Complete implementation (GitHub repository)

Pretrained model

Requirements.txt for reproduction

Documentation

Technical Report 6-8 pages

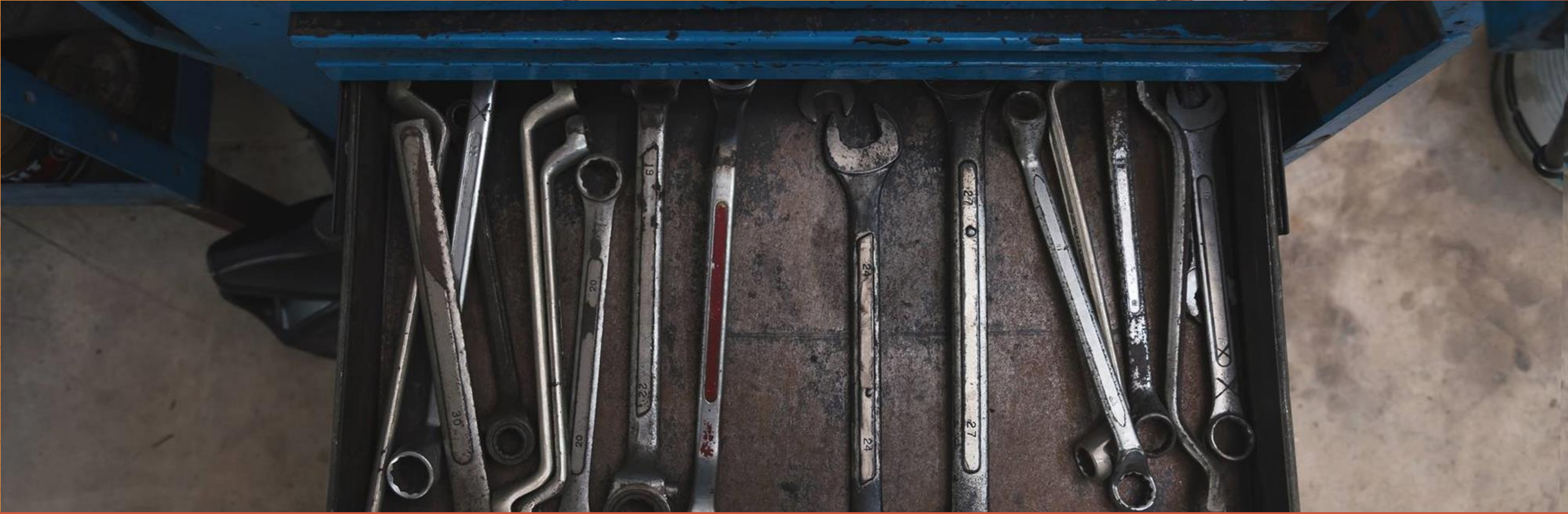
- Attention weight analysis

Result Visualizations

Additional Materials

README or user guide

This presentation



Q+A

We will now open the floor for questions.