ISTVT: Interpretable Spatial-Temporal Video Transformer for Deepfake Detection

Presented by:

Anuj Gour (220181)

Aryan Gautam (220222)

Naman Patidar (230679)

Khushal Karadiya (230558)

Our Goal: Impactful Deepfake Detection

Presenting "ISTVT: Interpretable Spatial-Temporal Video Transformer for Deepfake Detection"

Clear Visuals & Transitions

We use smooth, intuitive visuals and transitions to make complex concepts easy to understand and follow.

Real-World Relevance

Our research directly addresses urgent challenges posed by deepfakes in media, politics, and society, connecting technical advances to real-world impact.

Minimalist Design

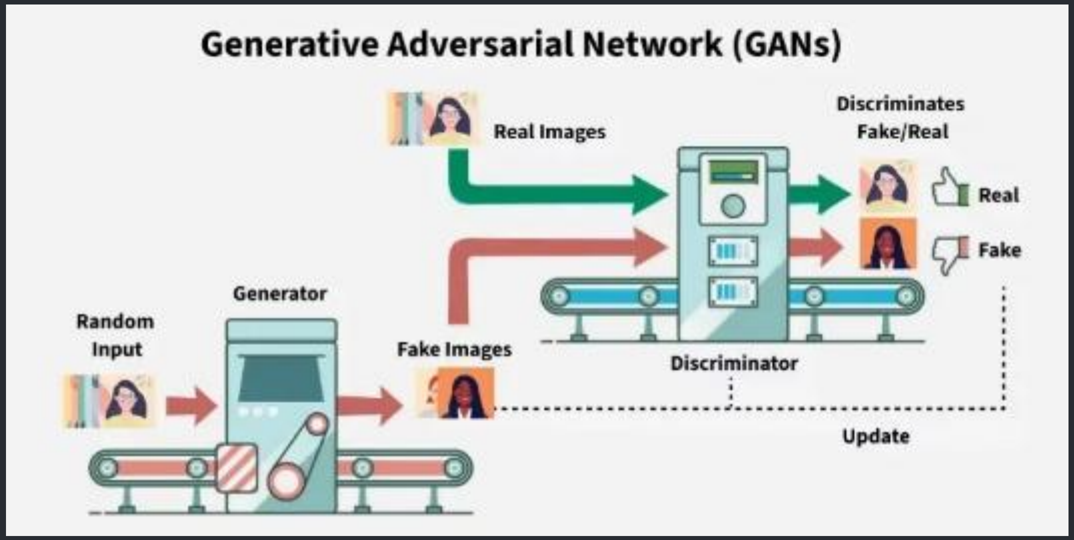
We focus on delivering key insights with clarity—prioritizing essential information over unnecessary text or clutter.

What Are Deepfakes?

Deepfakes are synthetic media—primarily videos—created using deep learning techniques, especially **Generative Adversarial Networks** (GANs), to replace or manipulate faces, voices, or actions in a realistic way.

🎥 Usually face-swapped or manipulated **videos and audio.**

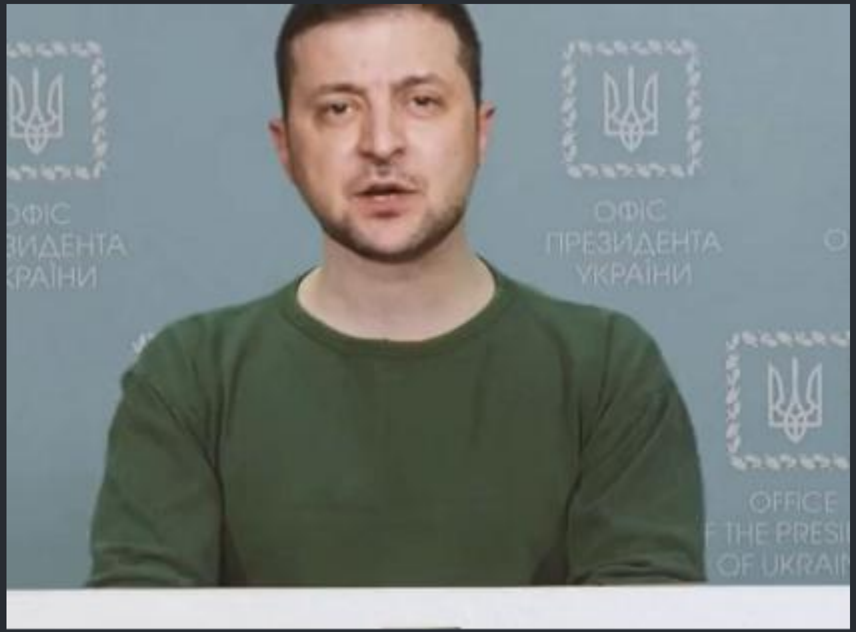
🤖 How?: Trained neural networks learn to generate highly realistic fake content.



Deepfakes Are Fooling the World...

In 2018, a fake video of Barack Obama circulated giving a speech he never made.

Click here



The 2022 Zelenskyy deepfake aimed to spread disinformation during conflict.

Click here

Indian PM Modi deepfakes have appeared, highlighting political manipulation risks.

Click here

Where Current Models Fall Short

We need a model that sees both space & time… and explains why.

Frame-based

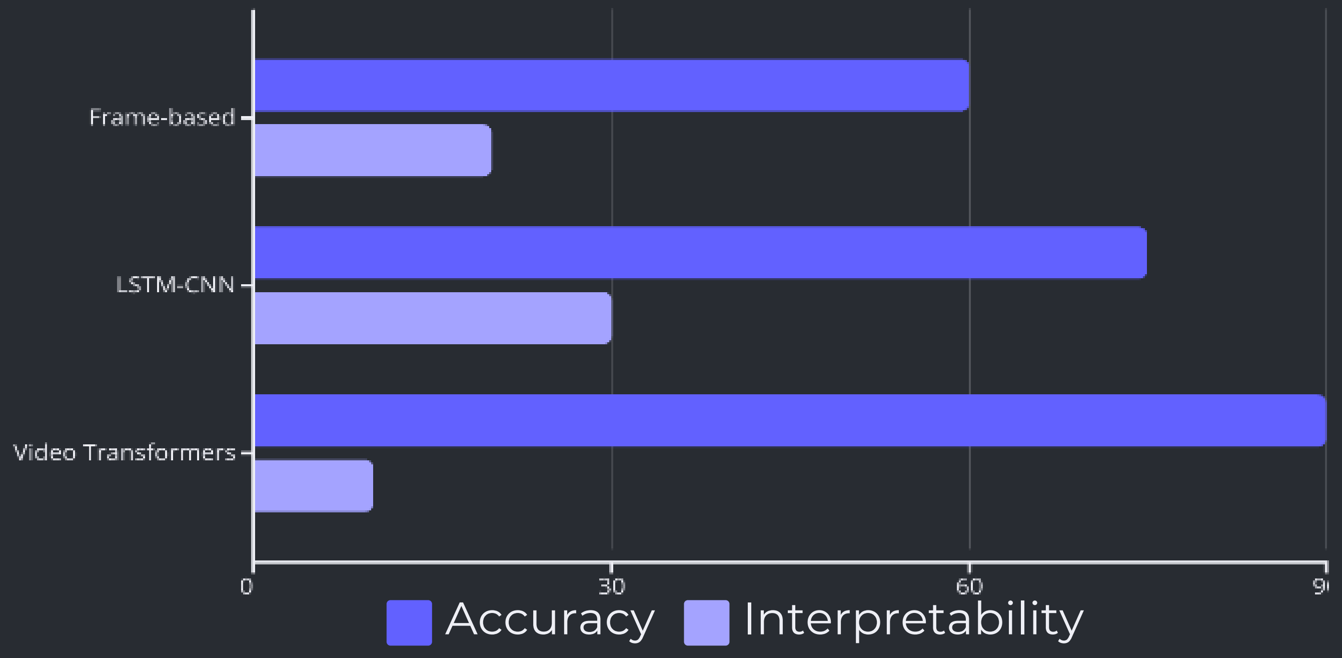
Poor on unseen fakes

Temporal Models

Can't isolate fine-grained artifacts

Transformers

Accurate but not interpretable



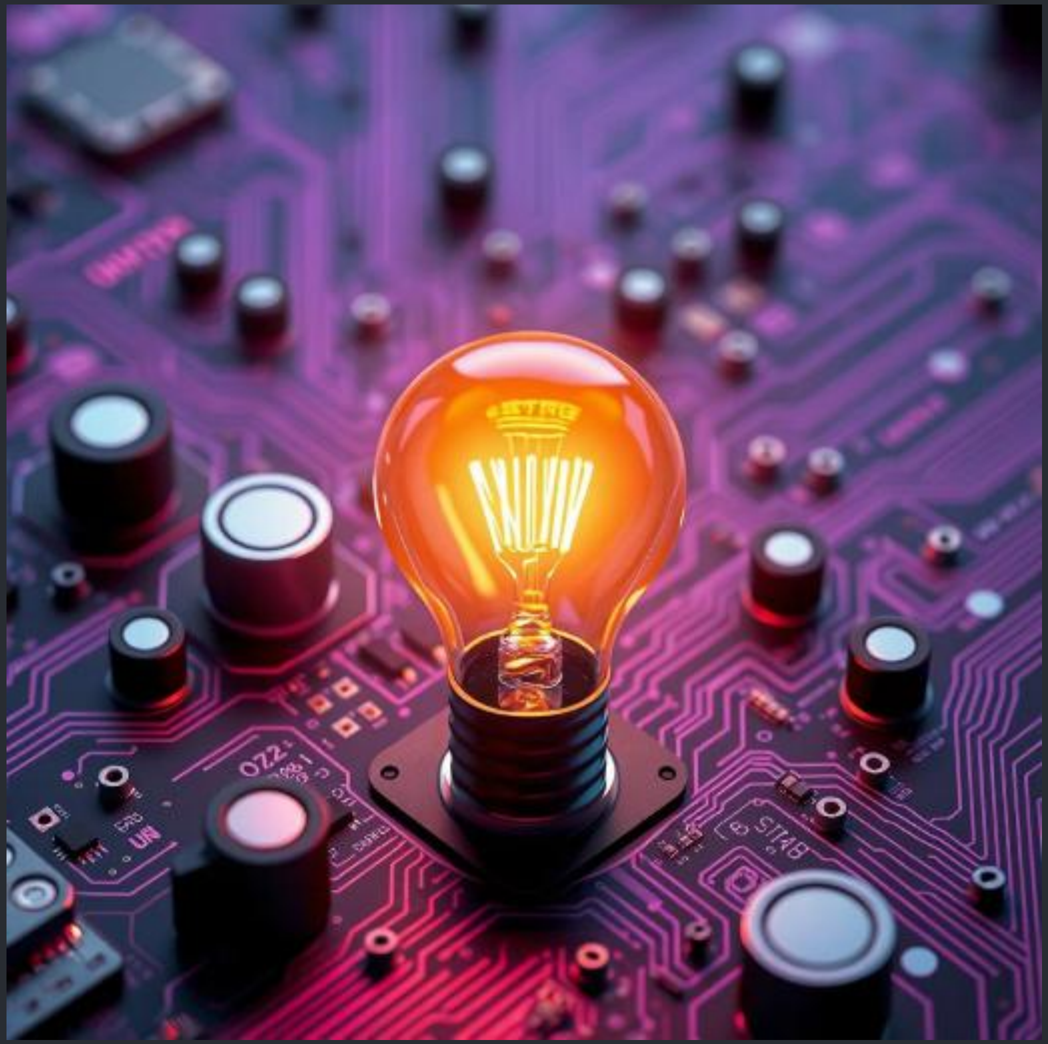
Introducing ISTVT 💡

Our new model, the Interpretable Spatial-Temporal Video Transformer (ISTVT), addresses the limitations of existing Deepfake detection methods.

Unlike frame-based CNNs, ISTVT captures **temporal clues**, enhancing detection on unseen fake videos.

Unlike Temporal models, ISTVT excels with **fine spatial details**, improving effectiveness on subtle, high-resolution forgeries.

Unlike traditional Transformers, ISTVT balances **high accuracy** with **interpretability**, allowing for detailed analysis.



Introducing ISTVT 💡

Spatial-Temporal Inconsistencies

Captures both space and time

Explainable Decisions

Heatmaps show reasoning

Superior Performance

🚀 **High Accuracy** across datasets (FaceForensics++, DFDC, Celeb-DF)

🧩 **Better Generalization** on unseen deepfake techniques

🔍 **Explainable AI** – trustworthy and transparent decisions

Self-Subtract Mechanism

- Highlights subtle changes across frames by subtracting adjacent frame features → sharpens temporal clues.

ISTVT Architecture

```python
def __getitem__(self, idx):
    face_path = self.face_paths[idx]
    faces = torch.load(face_path)  # shape: (T, 3, 224, 224)

    if self.transform:
        faces = torch.stack([self.transform(face) for face in faces])

    label = 1 if 'fake' in os.path.basename(face_path) else 0
    return faces, torch.tensor(label, dtype=torch.float32)
```

1. Input: Frame sequence of shape $T \times C_i \times H_i \times W_i$

```python
class XceptionBackbone(nn.Module):
    def __init__(self, freeze_stages=True):
        super().__init__()
        self.backbone = timm.create_model(
            'xception', pretrained=True, features_only=True
        )
        self.out_channels = self.backbone.feature_info[-1]['num_chs']  # usually 2048

        if freeze_stages:
            for param in self.backbone.parameters():
                param.requires_grad = False


            for name, module in list(self.backbone.named_children())[-2:]:
                for param in module.parameters():
                    param.requires_grad = True

    def forward(self, x):
        features = self.backbone(x)  # List of feature maps
        return features[-1]          # Final feature map (B*T, 2048, H, W)
```

2. Feature extraction using Xception blocks → Output shape: $T \times C \times H \times W$

```python
def forward(self, x):  # (B, T, C, H, W)
    B, T, C, H, W = x.shape
    HW = H * W
    x = x.view(B, T, C, HW).permute(0, 1, 3, 2)  # (B, T, HW, C)
    x = self.proj(x)
```

3. Flattened to tokens: $T \times HW \times C$

4. **Add:**

- **Spatial classification tokens**: $T \times 1 \times C$
- **Temporal classification tokens**: $1 \times (HW + 1) \times C$
- Final token input to the transformer should be:

(T + 1) × (HW + 1) × C

```python
spatial_cls = self.spatial_cls.expand(B, T, 1, -1)
x = torch.cat([spatial_cls, x], dim=2)  # (B, T, HW+1, C)

temporal_cls = self.temporal_cls.expand(B, 1, x.size(2), -1)
x = torch.cat([temporal_cls, x], dim=1)  # (B, T+1, HW+1, C)

pos_embed = torch.randn(T + 1, x.size(2), self.C_out, device=x.device)
x = x + self.pos_embed  # (B, T+1, HW+1, C) with learnable embeddings
return x
```

5.
- Add **learnable position embeddings**
- Pass through **M spatial-temporal transformer blocks**
- Final output token at $(0,0)(0,0)$(0,0) → **MLP** → **Binary prediction**

```python
self.pos_embed = nn.Parameter(
    torch.randn(1, num_frames + 1, patch_tokens + 1, self.C_out)
)  # (1, T+1, HW+1, C)
```

```python
self.encoder_blocks = nn.Sequential(*[
    DecomposedSTBlock(embed_dim, num_heads) for _ in range(depth)
])
```

```python
cls_token = encoded[:, 0, 0]              # (B, C)
logits = self.classifier(cls_token)       # (B, 1)
return logits
```

Architecture Overview

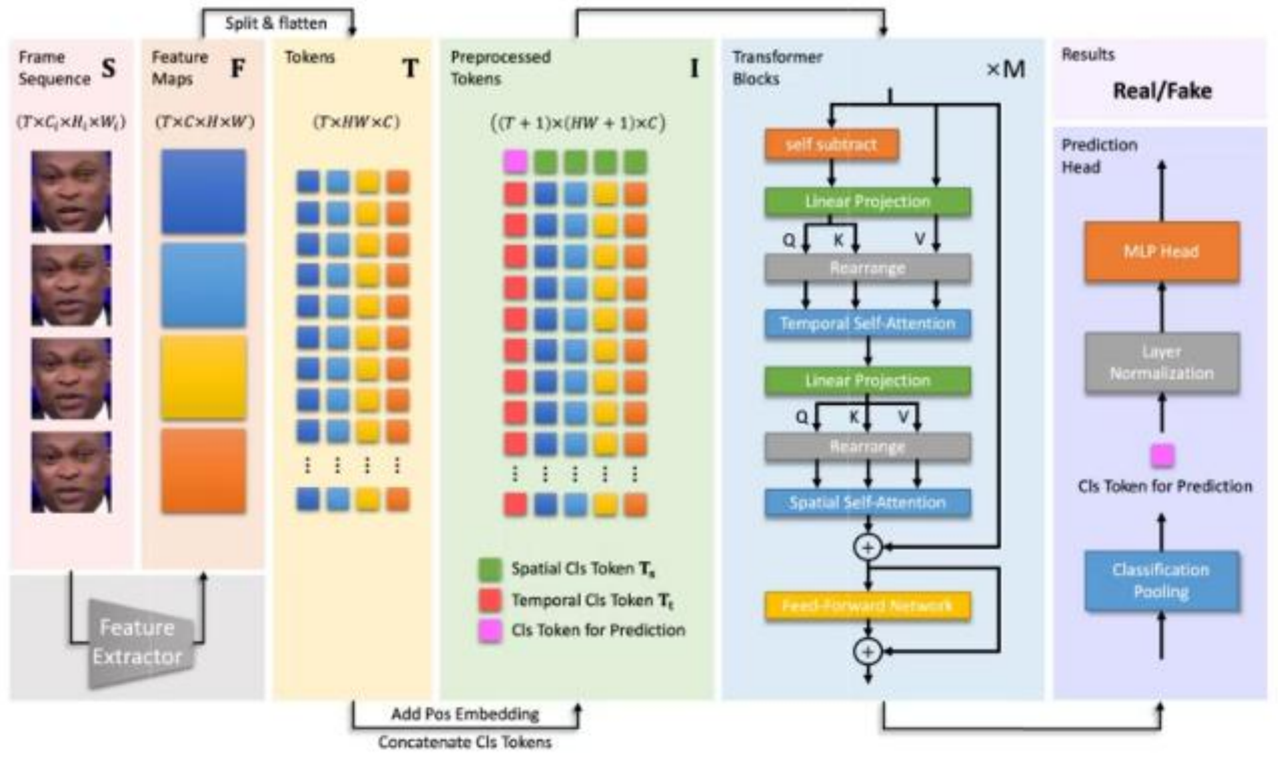


Fig. 2. The architecture of the proposed ISTVT. The feature extractor extracts the texture features from the input sequence and generates the feature maps. We split the feature maps into patches and flatten them to form the token sequence. The token sequence is then concatenated to the classification tokens and added with a position embedding. A spatial-temporal video transformer, consisting of several decomposed spatial-temporal blocks, takes the preprocessed tokens as inputs and outputs the results.

1

Xception

xtracts low-level spatial texture features from each frame (Entry flow of Xception used).

2

Patch Tokenization

Splits feature maps into spatial patches and flattens them into tokens

3

Spatial-Temporal Transformer

12 transformer blocks with decomposed Spatial & Temporal Self-Attention and a Self-Subtract mechanism.

4

MLP Classifier

Uses learned classification token for Real/Fake prediction via MLP head.

Separate Attention for Space and Time

Temporal Attention

Looks across same patch over time.

Operates over **temporal dimension** (time/frames)

Captures **motion dynamics** and **temporal consistency**

Spatial Attention

Looks at all patches in same frame

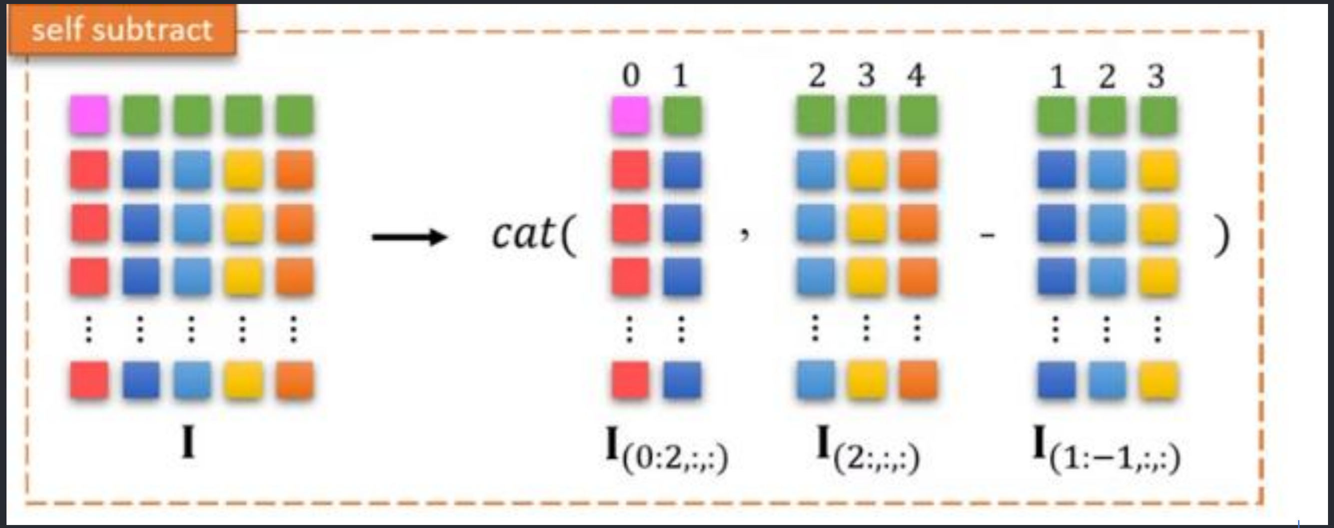
Operates over **spatial dimensions** (height × width)

Captures **scene structure** and **spatial context**

This separation significantly improves interpretability, allowing precise identification of manipulation.

Self Subtract Mechanism:

```python
def _self_subtract(self, x):
    return x - x.mean(dim=-1, keepdim=True)
```



The Self-Subtract Mechanism computes the difference between adjacent frame tokens to highlight temporal inconsistencies and suppress redundant static features before applying temporal self-attention.

```python
spatial_out = []
for t in range(T1):
    x_t = x[:, t]  # (B, P, C)
    q = k = self._self_subtract(x_t)
    out, _ = self.spatial_attn(q, k, x_t)
    spatial_out.append(self.norm1(x_t + out))
x = torch.stack(spatial_out, dim=1)  # (B, T+1, P, C)
```

```python
temporal_out = []
for p in range(P1):
    x_p = x[:, p]  # (B, T+1, C)
    q = k = self._self_subtract(x_p)
    out, _ = self.temporal_attn(q, k, x_p)
    temporal_out.append(self.norm2(x_p + out))
x = torch.stack(temporal_out, dim=1)  # (B, P, T+1, C)
x = x.permute(0, 2, 1, 3)  # (B, T+1, P, C)
return x
```

1. **Highlights Temporal Artifacts**Subtracts adjacent frame tokens to focus on **inter-frame distortions**.

2. **Removes Redundancy, Retains Details**Suppresses **unchanging features** while preserving important **spatial cues** using original tokens for value projection.

3. **Boosts Detection Robustness**Enhances model focus on meaningful temporal changes → better generalization and performance.

Mathematics of Decomposed Attention

Temporal Attention (at spatial index j):

$$O_t(:, j, :, :) = \mathrm{softmax}(\frac{Q(:, j)K(:, j)^T}{\sqrt{D}})V(:, j)$$

Spatial Attention (at temporal index k):

$$O_s(k, :, :, :) = \mathrm{softmax}(\frac{Q(k, :)K(k, :)^T}{\sqrt{D}})V(k, :)$$

Reduced Attention Complexity:

Reduces attention complexity: $O(T^2H^2W^2) \rightarrow O(T^2 + H^2W^2)$

How ISTVT Explains Its Decisions

Highlights Suspicious Features

Identifies unnatural textures, asymmetries, or distortions often missed by humans.

Detects Abnormal Motion

eveals subtle inter-frame motion anomalies introduced by frame-wise manipulation.

1 Layer-wise Relevance Propagation (LRP)

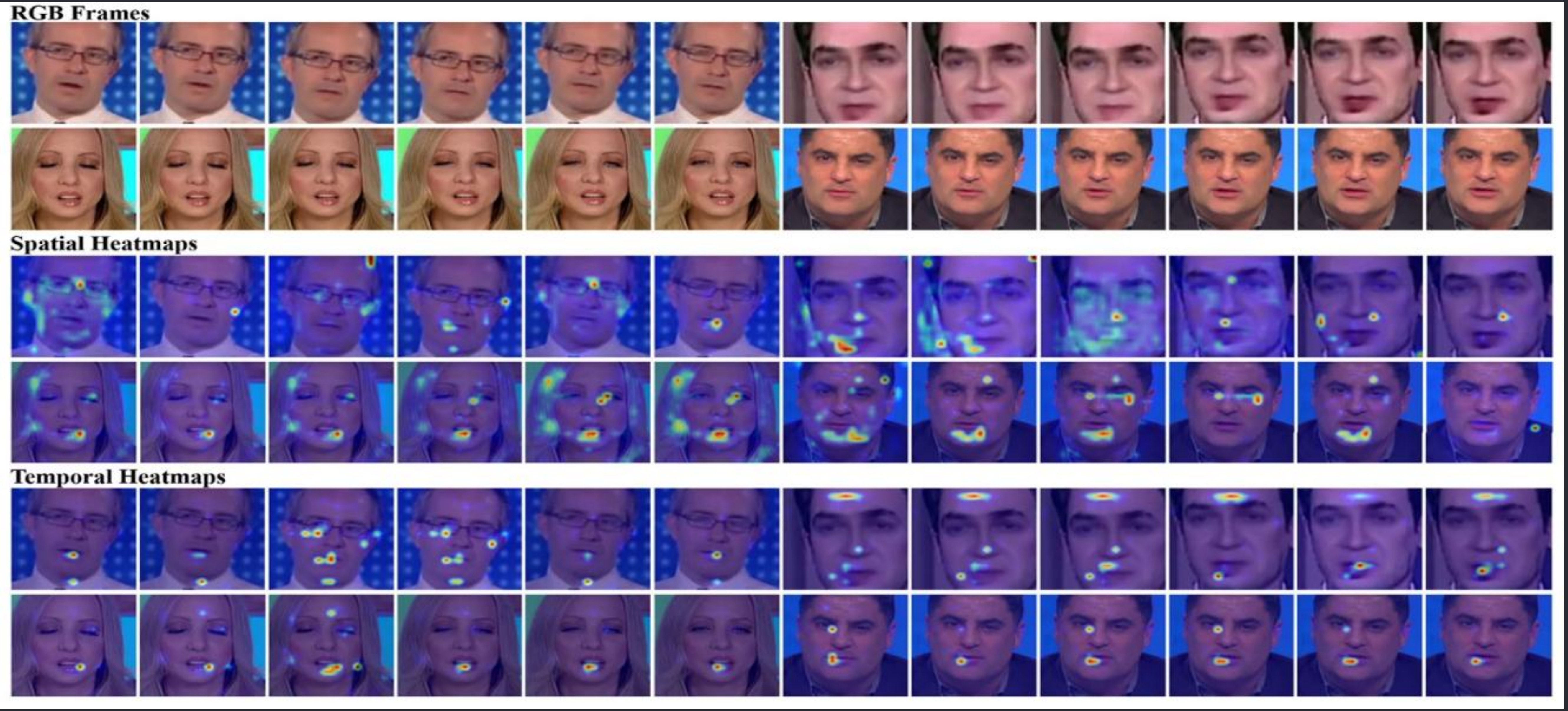
Explains model reasoning

$$\bar{A}_d^{(m)}(i,:,:) = I + \max(E_h(R_d^{(m)}(:,i,:,:) \circ \nabla A_d^{(m)}(:,i,:,:)),0)$$

2 Final Heatmap Calculation

$$Ud(i,:,:) = \prod_{m=1}^{M} \bar{A}_d^{(m)}(i,:,:)$$

Visualisation of Spatial and Temporal Heatmaps



Spatial Heatmap: Highlights specific regions within a frame that show anomalies or manipulations, such as unnatural textures or facial distortions.
Temporal Heatmap: Visualizes inter-frame inconsistencies and sudden changes, pinpointing where motion anomalies or temporal artifacts occur.

🧪 Experimental Setup

Datasets

- **FaceForensics++:** 1000 real + fake videos from 4 manipulation methods
- **Celeb-DF:** 590 real, 5639 high-quality fake videos
- **DFDC:** Preview set from Deepfake Detection Challenge (2020)
- **FaceShifter, DeeperForensics:** High-quality fakes built on FaceForensics++

Preprocessing & Training Details

- Face detection: MTCNN
- Alignment: Based on nose landmark
- Resize: 300×300 facial crops
- Sequence: 6 continuous aligned frames
- 270 frames used per video (FaceForensics++)

Training Details Cont.

- Feature Extractor: Entry flow of Xception
- Transformer: 12 blocks, 8 heads
- Optimizer: SGD with warm-up, LR = 0.0005
- Hardware: 4× Tesla V100 GPUs, 100 epochs
- Selection: Best model via validation accuracy

Performance

1) Intra-Dataset Performance (Same dataset for training and testing):

- **Datasets Used**: FaceForensics++, Celeb-DF, DFDC.
- **ISTVT** significantly outperforms other models like ViViT, VidTr, VTN, and CNN-based Xception (XN-avg).
- CNN-based methods like VidTr often miss fine temporal clues.
- **ISTVT captures both spatial & temporal artifacts**, boosting accuracy.

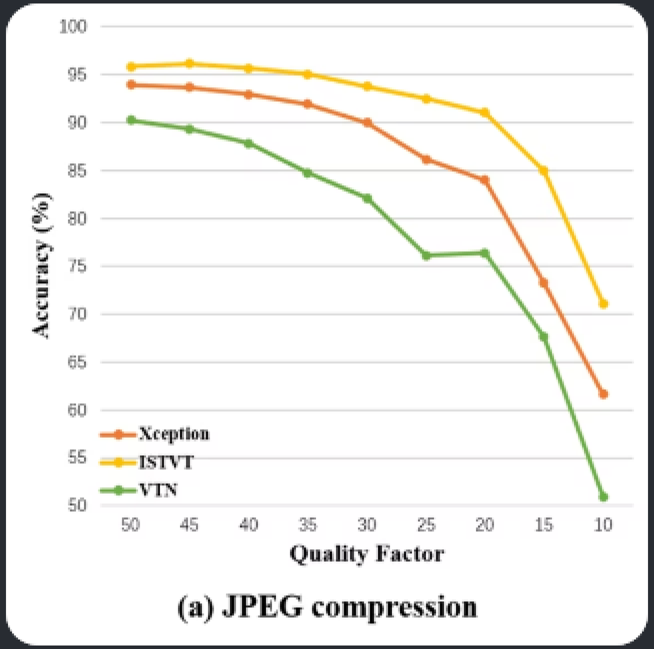
Performance

2) Cross-Dataset Performance (Trained on FF++, tested on others):

- **Evaluation Metric**: AUROC.
- ISTVT shows **strong generalization**, especially on difficult datasets like DFDC.
- Compared to FTCN (which uses longer sequences), ISTVT performs better in:
    - Complex conditions (e.g., varying lighting/head pose).
    - Interpretability (clear separation of spatial/temporal focus).
    - Intra-dataset accuracy.

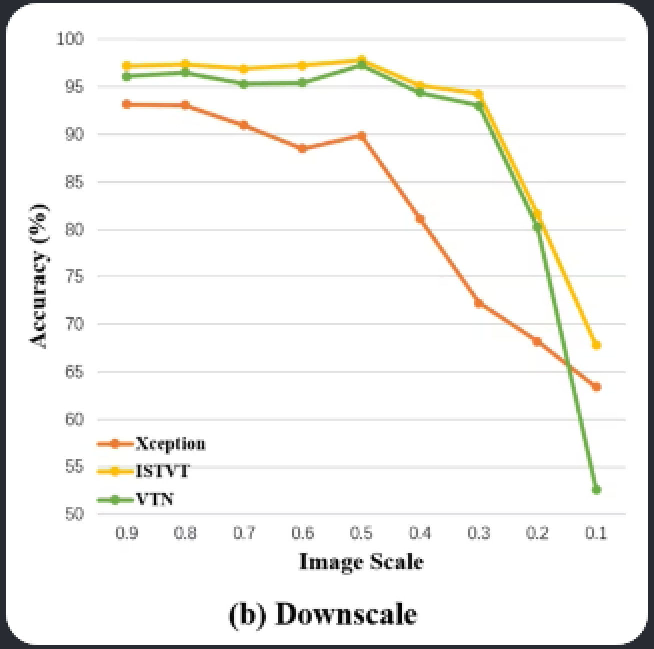
Robustness to Perturbations

Evaluating Model Performance on Noisy Deepfake Inputs



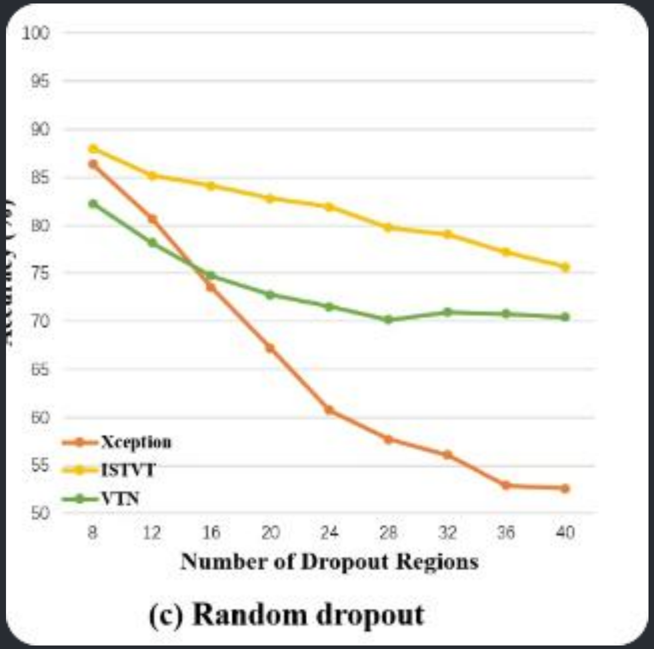
JPEG Compression

ISTVT maintains strong performance, outperforming baselines due to its unique attention and self-subtract mechanism.



Downscaling

ISTVT robustly performs at low image scales by capturing subtle inter-frame inconsistencies, surpassing Xception and VTN.



Random Dropout

ISTVT handles missing regions well, leveraging temporal artifacts, while Xception fails under high dropout.

ISTVT consistently demonstrates superior robustness against real-world video perturbations, validating its advanced architecture.

Conclusion

ISTVT effectively identifies deepfakes by analysing spatial and temporal inconsistencies, demonstrating efficiency and robustness across diverse datasets and video qualities.

Future Work

- Enhance generalisation for emerging deepfake techniques.
- Integrate multi-modal analysis (audio, text, video) for comprehensive detection.
- Create lightweight, real-time models suitable for mobile and edge device deployment.
- Strengthen fairness and minimise bias in deepfake detection outcomes.

Thank You !

We hope you found our work on ISTVT valuable.

We welcome any questions or discussions.