AI VIET NAM
@aivietnam.edu.vn

Video Classification - Project

TA Minh-Duc Bui

Outline

1. Overview of Video Data
2. RWF2000 Dataset for Violence Detection Task
3. Models for Video Classification Task
4. Optional: Self-Supervised Learning for Video Data

Outline

1. Overview of Video Data
2. RWF2000 Dataset for Violence Detection Task
3. Models for Video Classification Task
4. Optional: Self-Supervised Learning for Video Data

AI VIET NAM
@aivietnam.edu.vn

Video Classification Overview

Video = 2D + Time

A video is a **sequence** of images
4D tensor:
T x 3 x H x W

Video Classification Overview

Input video:
T x 3 x H x W

Swimming

Running

Jumping

Standing

Video Classification Overview

images: recognize **objects**

Dog
Cat
Fish
Bird

videos: recognize **actions**

Swimming
Running
Jumping
Standing

Video Classification Overview

Videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of **uncompressed video**
(3 bytes per pixel):

- SD (640 x 480): ~1.5 GB per minute
- HD (1920 x 1080): ~10 GB per minute

Solution: train on short **clips**: low fps and low spatial resolution
e.g. T = 16, H=W=112
(3.2 seconds at 5 FPS, 588 KB)

Video Classification Overview

Original video: long, high FPS

Training: Train model to classify short **clips** with low FPS

Testing: Run model on different clips, average predictions

Outline

1. Overview of Video Data
2. RWF2000 Dataset for Violence Detection Task
3. Models for Video Classification Task
4. Optional: Self-Supervised Learning for Video Data

RWF2000 - Violence Detection

Collected raw surveillance videos from YouTube, sliced them into clips within **5s at 30 fps**, and labeled each clip as Violent or Non-Violent.

Finally, we have **2000 clips** and **300,000 frames** as a new data set for detecting real-world violent behavior under a surveillance camera.

Challenges

Since all the videos are captured by surveillance cameras in public places, many may need better imaging quality due to dark environments, fast movement of objects, lighting blur, etc.

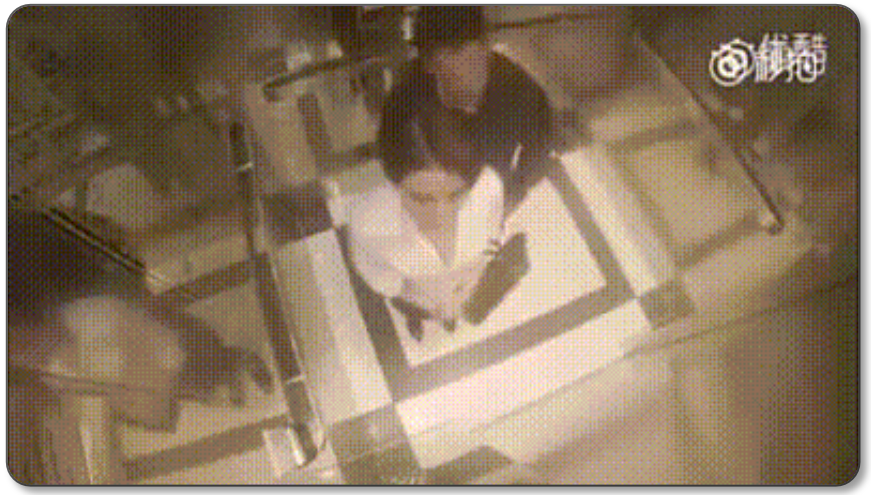
Cheng, Ming, Kunjing Cai, and Ming Li. "RWF-2000: an open large scale video database for violence detection." 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

RWF2000 - Violence Detection

Only part of the person appears in the picture

Crowds and chaos

Challenges



Transient Action

Low resolution

Cheng, Ming, Kunjing Cai, and Ming Li. "RWF-2000: an open large scale video database for violence detection." 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

Create Video Dataset

```python
class VideoDataset(Dataset):
    def __init__(self, root_dir, phase="train", transform=None, n_frames=None):
        """
        Args:
            root_dir (string): Directory with all the videos
              (each video as a subdirectory of frames).
            transform (callable, optional): Optional transform to be applied on a sample.
            n_frames (int, optional): Number of frames to sample from each video, uniformly.
              If None, use all frames.
        """
        self.root_dir = root_dir
        self.transform = transform
        self.n_frames = n_frames
        self.phase = phase
        self.videos, self.labels = self._load_videos()

    def _load_videos(self):
        videos, labels = [], []

        video_folers = os.listdir(
            os.path.join(self.root_dir, self.phase)
        )  # NonFight, Fight

        # Loop through each class
        class_id = 0
        for folder in video_folers:
            # all videos in the same class
            all_video_paths = os.listdir(
                os.path.join(self.root_dir, self.phase, folder)
            )

            # load frames from each video
            for video_path in all_video_paths:
                frames = []
                video_folder = os.path.join(
                    self.root_dir, self.phase, folder, video_path
                )
                for frame in os.listdir(video_folder):
                    frames.append(os.path.join(video_folder, frame))
                if self.n_frames:
                    frames = self._uniform_sample(frames, self.n_frames)
                videos.append(frames)
                labels.append(class_id)

            class_id += 1

        return videos, labels
```

```python
    def _uniform_sample(self, frames, n_frames):
        """
        Helper method to uniformly sample n_frames from the frames list.
        """
        stride = max(1, len(frames) // n_frames)
        sampled = [frames[i] for i in range(0, len(frames), stride)]
        return sampled[:n_frames]

    def __len__(self):
        return len(self.videos)

    def __getitem__(self, idx):
        video_frames = self.videos[idx]
        label = self.labels[idx]
        images = []
        for frame_path in video_frames:
            image = Image.open(frame_path).convert("RGB")
            if self.transform:
                image = self.transform(image)
            images.append(image)

        # Stack images along new dimension (sequence length)
        data = torch.stack(images, dim=0)

        # Rearrange to have the shape (C, T, H, W)
        data = data.permute(1, 0, 2, 3)
        return data, label
```

```
- dataset/rwf-2000
  - train
    - Fight
      - video1
        - frame1.jpg
          ..
        - frame30.jpg
        ...
      - videoN
    - NonFight
  - val
    - Fight
    - NonFight
```

Create Video Dataset

```python
BATCH_SIZE = 16
MAX_LEN = 16
IMAGE_SIZE = 224

transform = transforms.Compose(
    [
        transforms.Resize((IMAGE_SIZE, IMAGE_SIZE)),
        transforms.ToTensor(),
    ]
)

# Load dataset
train_dataset = VideoDataset(root_dir='./dataset/rwf-2000', phase="train",
                             transform=transform, n_frames=MAX_LEN)

val_dataset = VideoDataset(root_dir='./dataset/rwf-2000',phase="val",
                           transform=transform, n_frames=MAX_LEN)

# Count number of cpus
cpus = os.cpu_count()
print(f"Number of cpus: {cpus}")

# Create data loaders
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE,
                          num_workers=cpus, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE,
                        num_workers=cpus, shuffle=False)
```

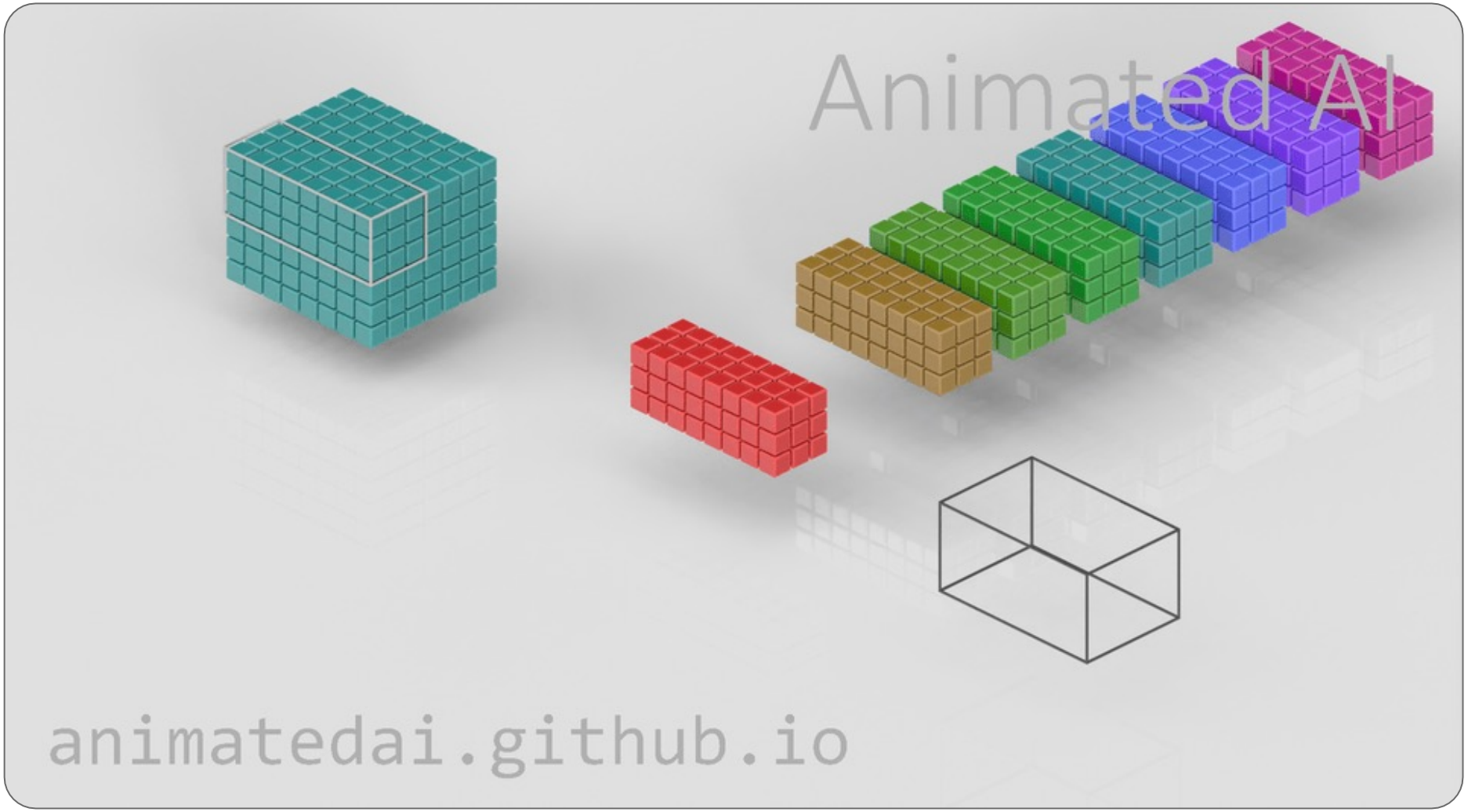
Outline

1. Overview of Video Data
2. RWF2000 Dataset for Violence Detection Task
3. Models for Video Classification Task
4. Optional: Self-Supervised Learning for Video Data

AI VIET NAM
@aivietnam.edu.vn

From 2D to 3D CNNs

What is this?
2D or 3D CNN?

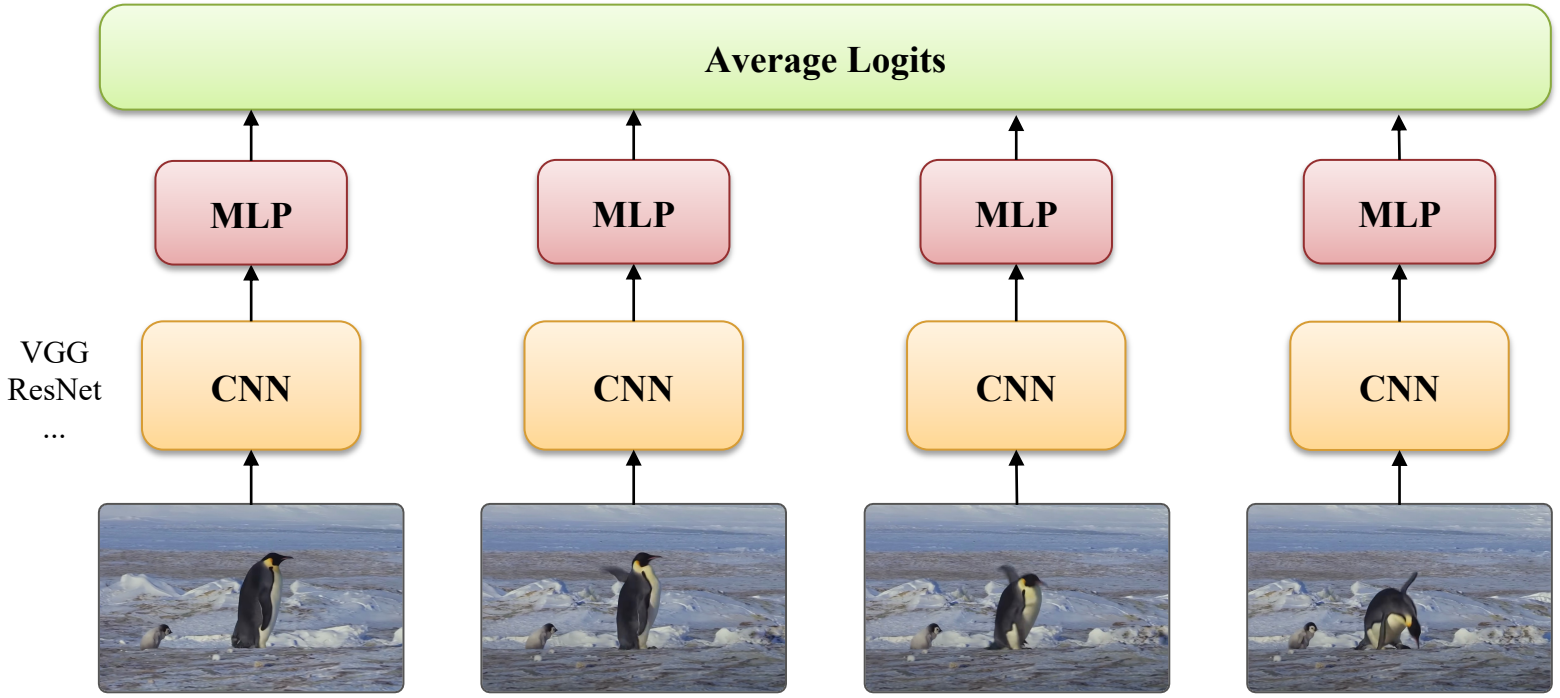


https://animatedai.github.io/

Video Classification: Single-Frame CNN

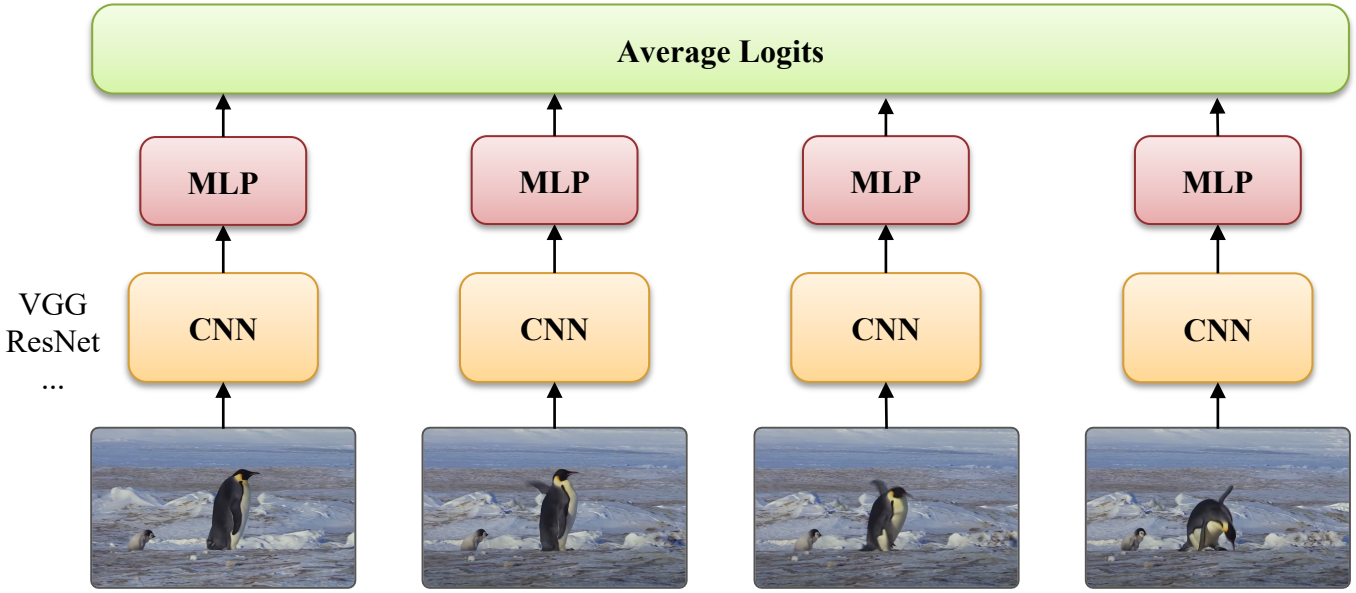
- **Simple idea**: train normal 2D CNN to classify video frames independently! (Average predicted probs at test-time)
- Often **a solid baseline** for video classification.

Run 2D CNN on each frame, and feed to MLP



Problem: pooling is not aware of the temporal order!

Video Classification: Single-Frame CNN




```python
from torchvision.models import resnet18

class Model(nn.Module):
    def __init__(self, num_classes=2):
        super(Model, self).__init__()
        self.resnet = resnet18(pretrained=True)
        self.resnet.fc = nn.Sequential(
          nn.Linear(self.resnet.fc.in_features, 512)
        )
        self.fc1 = nn.Linear(512, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x_3d):
        # (bs, C, T, H, W) → (bs, T, C, H, W)
        x_3d = x_3d.permute(0, 2, 1, 3, 4)

        logits = []
        for t in range(x_3d.size(1)):
            out = self.resnet(x_3d[:, t, :, :, :])

            x = self.fc1(out)
            x = F.relu(x)
            x = self.fc2(x)

            logits.append(x)

        # mean pooling
        logits = torch.stack(logits, dim=0)
        logits = torch.mean(logits, dim=0)
        return logits
```

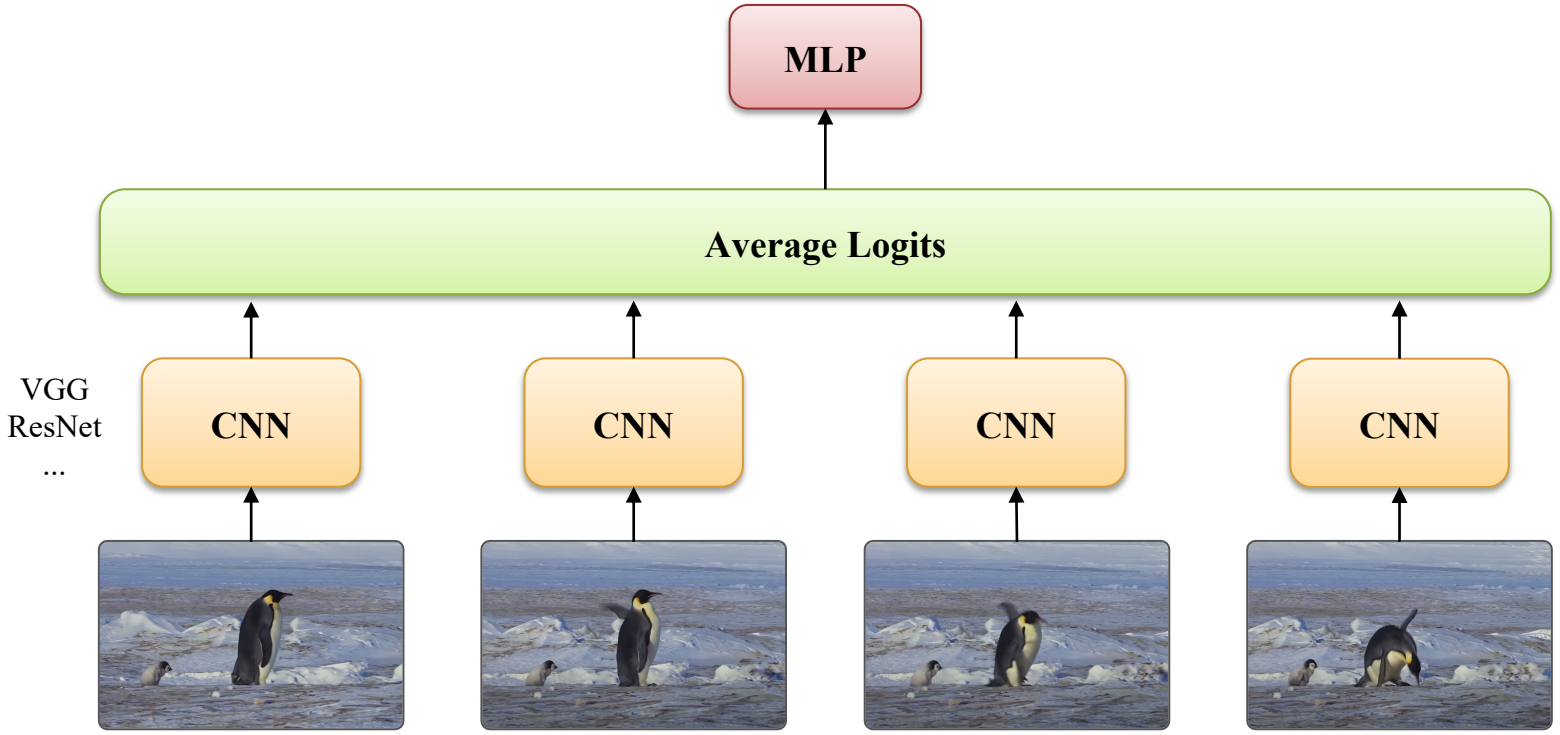
Video Classification: Single-Frame CNN

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8

Video Classification: Late Fusion

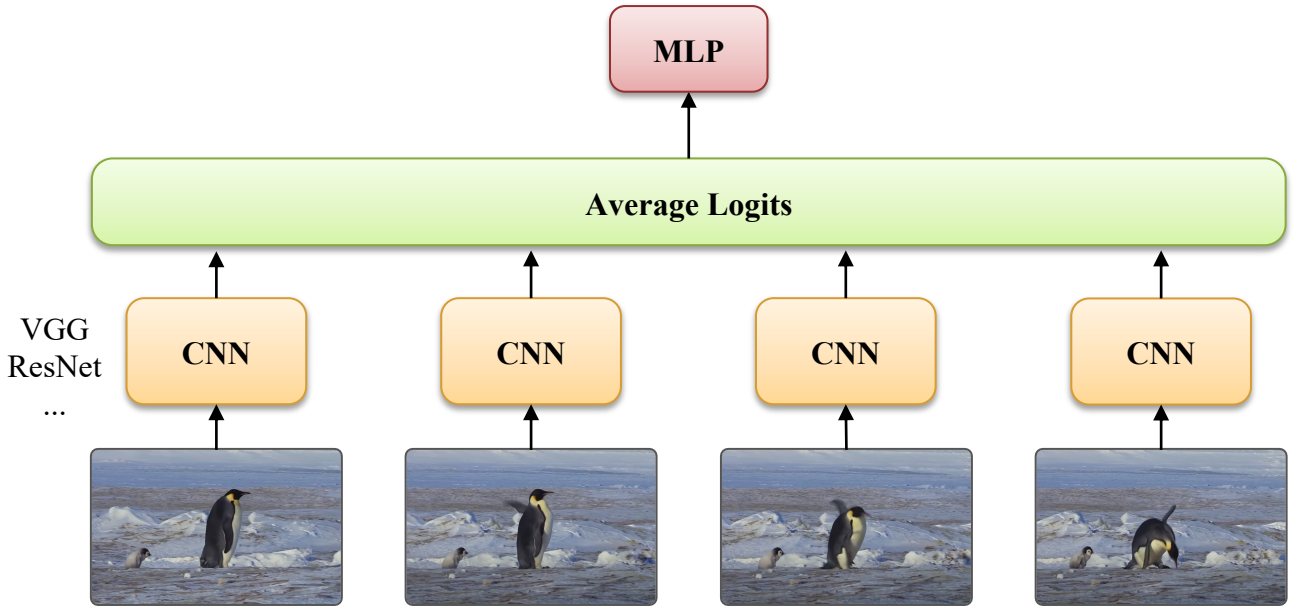
Intuition: Get a high-level appearance of each frame, and combine them

Run 2D CNN on each frame, concatenate features, and feed to MLP



Problem: Hard to compare low-level motion between frames

Video Classification: Late Fusion




```python
from torchvision.models import resnet18

class Model(nn.Module):
    def __init__(self, num_classes=2):
        super(Model, self).__init__()
        self.resnet = resnet18(pretrained=True)
        self.resnet.fc = nn.Sequential(
          nn.Linear(self.resnet.fc.in_features, 512)
        )
        self.fc1 = nn.Linear(512, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x_3d):
        # (bs, C, T, H, W) → (bs, T, C, H, W)
        x_3d = x_3d.permute(0, 2, 1, 3, 4)

        features = []
        for t in range(x_3d.size(1)):
            out = self.resnet(x_3d[:, t, :, :, :])
            features.append(out)

        # average pooling
        out = torch.mean(torch.stack(features), 0)

        x = self.fc1(out)
        x = F.relu(x)
        x = self.fc2(x)
        return x
```

Video Classification: Late Fusion

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0

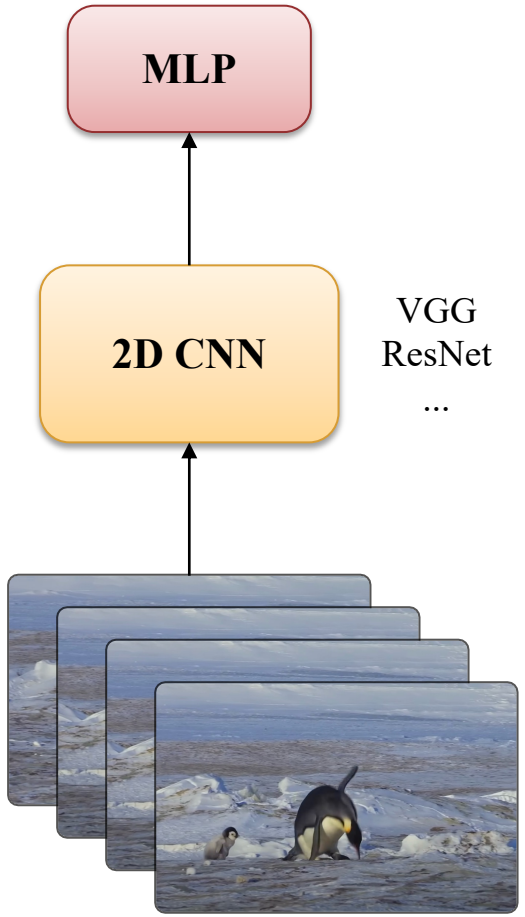
Video Classification: Early Fusion

Intuition: Get a high-level appearance of each frame and combine them

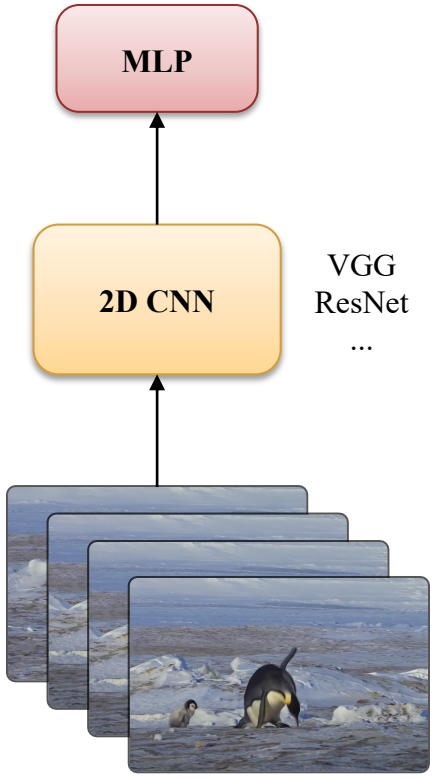
First 2D convolution collapses all temporal information:

- **Input**: 3T x H x W
- **Output**: D x H x W

Problem: Hard to compare low-level motion between frames



Video Classification: Early Fusion




```python
from torchvision.models import resnet18

class Model(nn.Module):
    def __init__(self, num_classes=2, num_input_channel=48):
        super(Model, self).__init__()
        self.resnet = resnet18(pretrained=True)
        self.resnet.conv1 = nn.Conv2d(
            num_input_channel, 64, kernel_size=7,
            stride=2, padding=3, bias=False
        )
        self.resnet.fc = nn.Sequential(
            nn.Linear(self.resnet.fc.in_features, 512)
        )
        self.fc1 = nn.Linear(512, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x_3d):
        # (bs, C, T, H, W)
        # concatenate all C and T dimensions to make it (bs, C*T, H, W)
        x_3d = x_3d.permute(0, 2, 1, 3, 4).contiguous()
        x_3d = x_3d.view(x_3d.size(0), x_3d.size(1) * x_3d.size(2),
                         x_3d.size(3), x_3d.size(4))

        out = self.resnet(x_3d)

        x = self.fc1(out)
        x = F.relu(x)
        x = self.fc2(x)
        return x
```

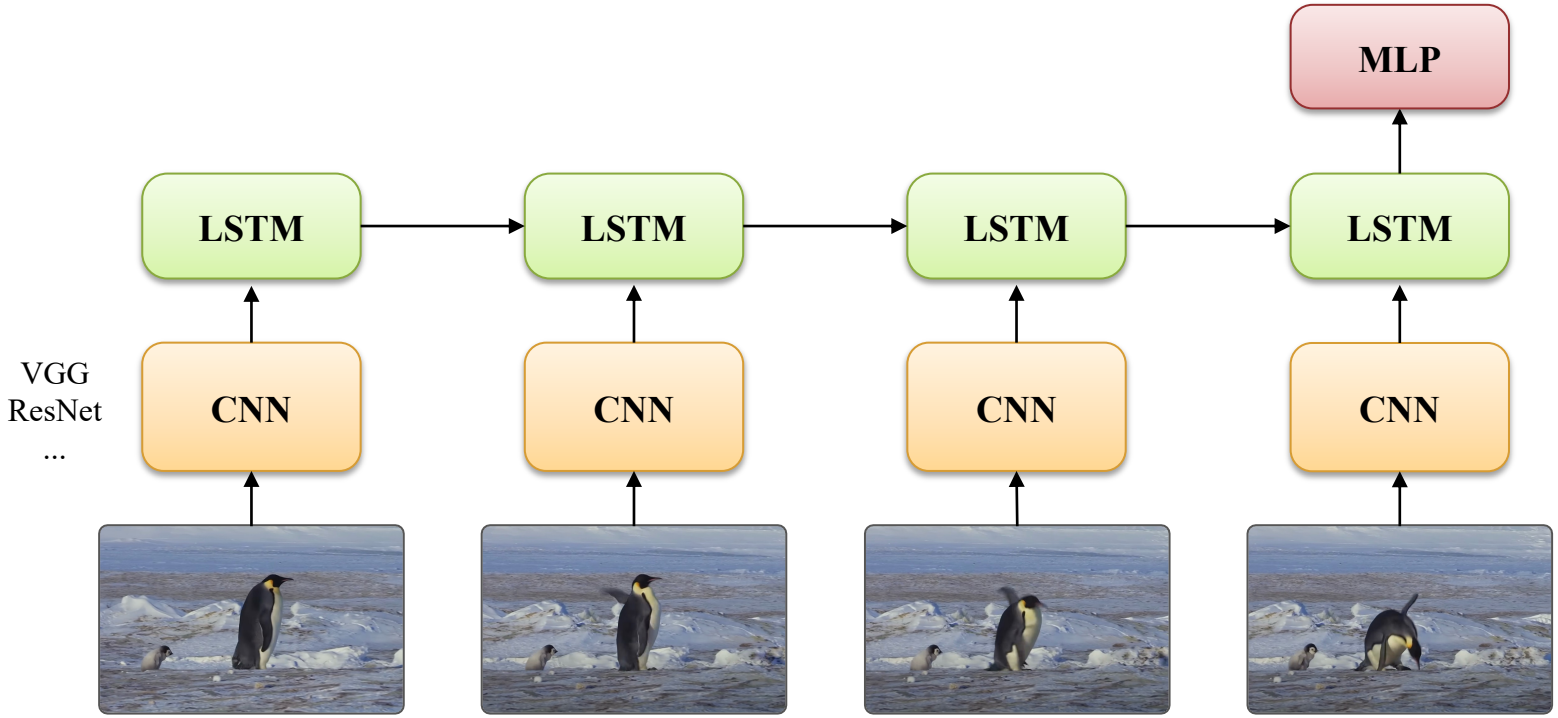
Video Classification: Early Fusion

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3

Video Classification: CNN-LSTM

- **Intuition**: Get a high-level appearance of each frame, and combine them
- **Recurrent Neural Networks** are well suited for processing sequences.

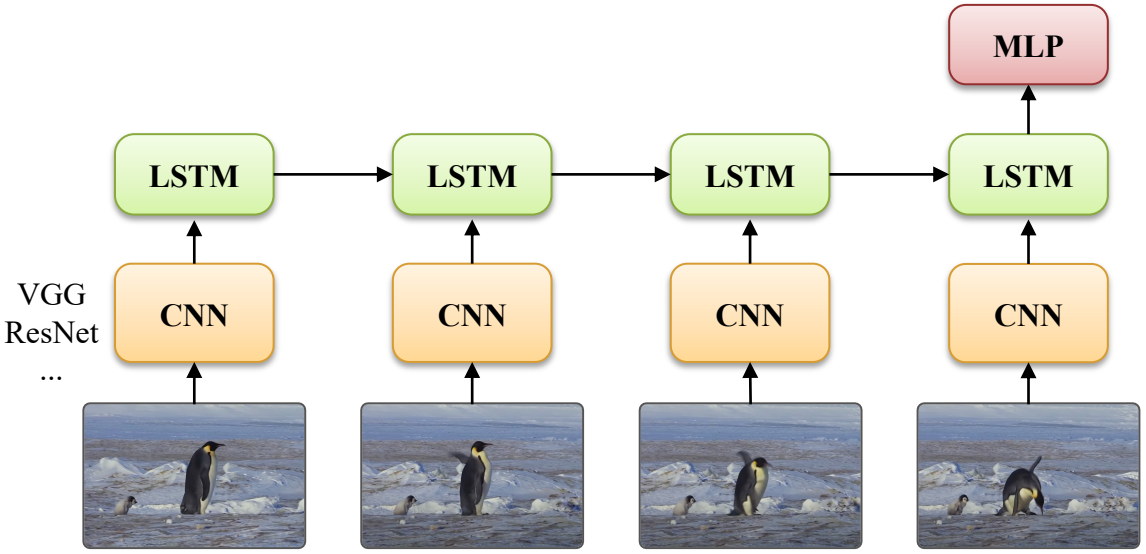
Run 2D CNN on each frame, concatenate features, and feed to LSTM



Problem: RNNs are sequential and cannot be parallelized.

AI VIET NAM
@aivietnam.edu.vn

Video Classification: CNN-LSTM




```python
from torchvision.models import resnet18

class Model(nn.Module):
    def __init__(self, num_classes=2):
        super(Model, self).__init__()
        self.resnet = resnet18(pretrained=True)
        self.resnet.fc = nn.Sequential(
            nn.Linear(self.resnet.fc.in_features, 512)
        )
        self.lstm = nn.LSTM(input_size=512, hidden_size=389, num_layers=3)
        self.fc1 = nn.Linear(389, 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x_3d):
        # (bs, C, T, H, W) → (bs, T, C, H, W)
        x_3d = x_3d.permute(0, 2, 1, 3, 4)

        hidden = None
        for t in range(x_3d.size(1)):
            with torch.no_grad():
                x = self.resnet(x_3d[:, t, :, :, :])
            out, hidden = self.lstm(x.unsqueeze(0), hidden)

        x = self.fc1(out[-1, :, :])
        x = F.relu(x)
        x = self.fc2(x)
        return x
```

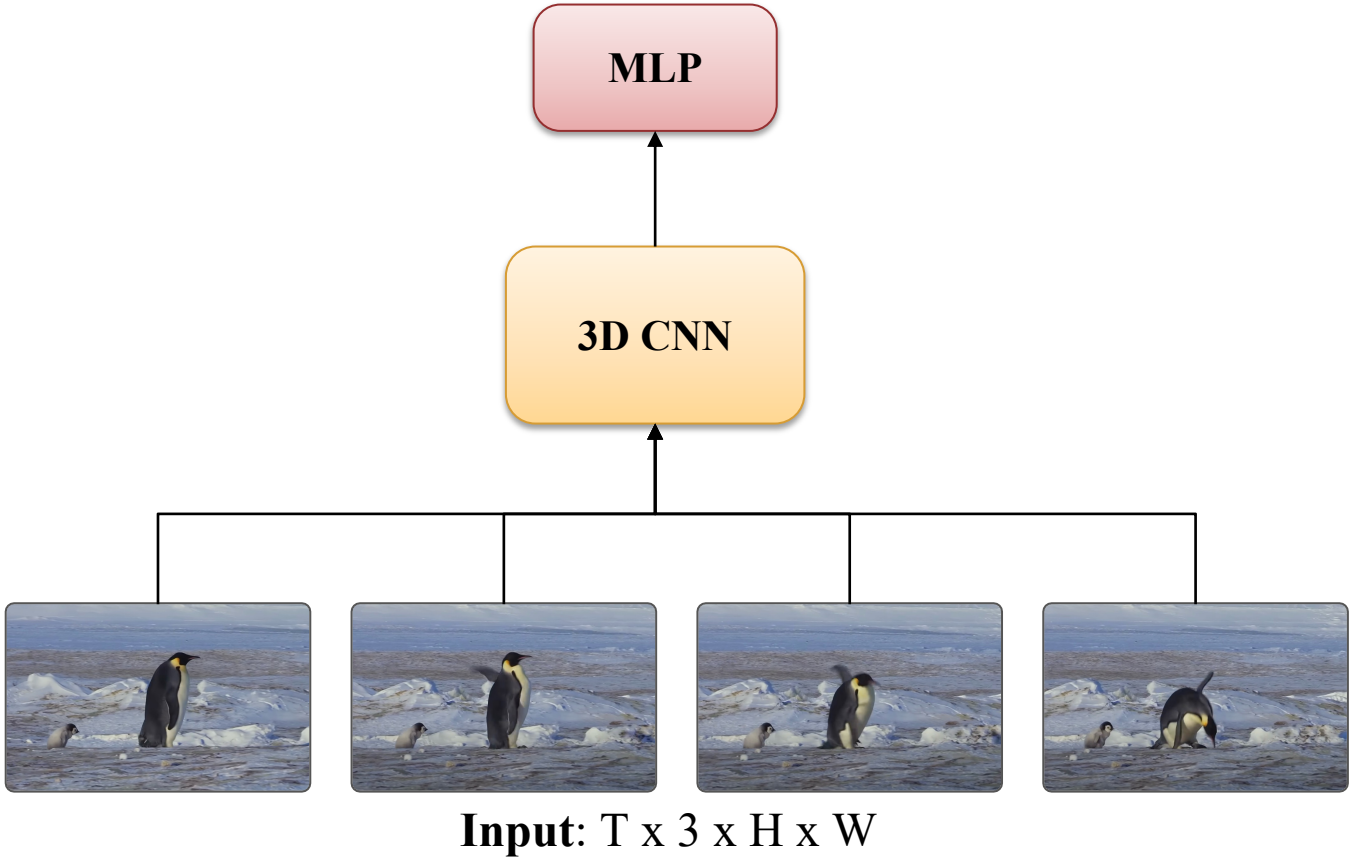
Video Classification: CNN-LSTM

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3
CNN-LSTM	15.3M	63.0
CNN-Mamba	16.4M	83.0

Video Classification: 3D CNN

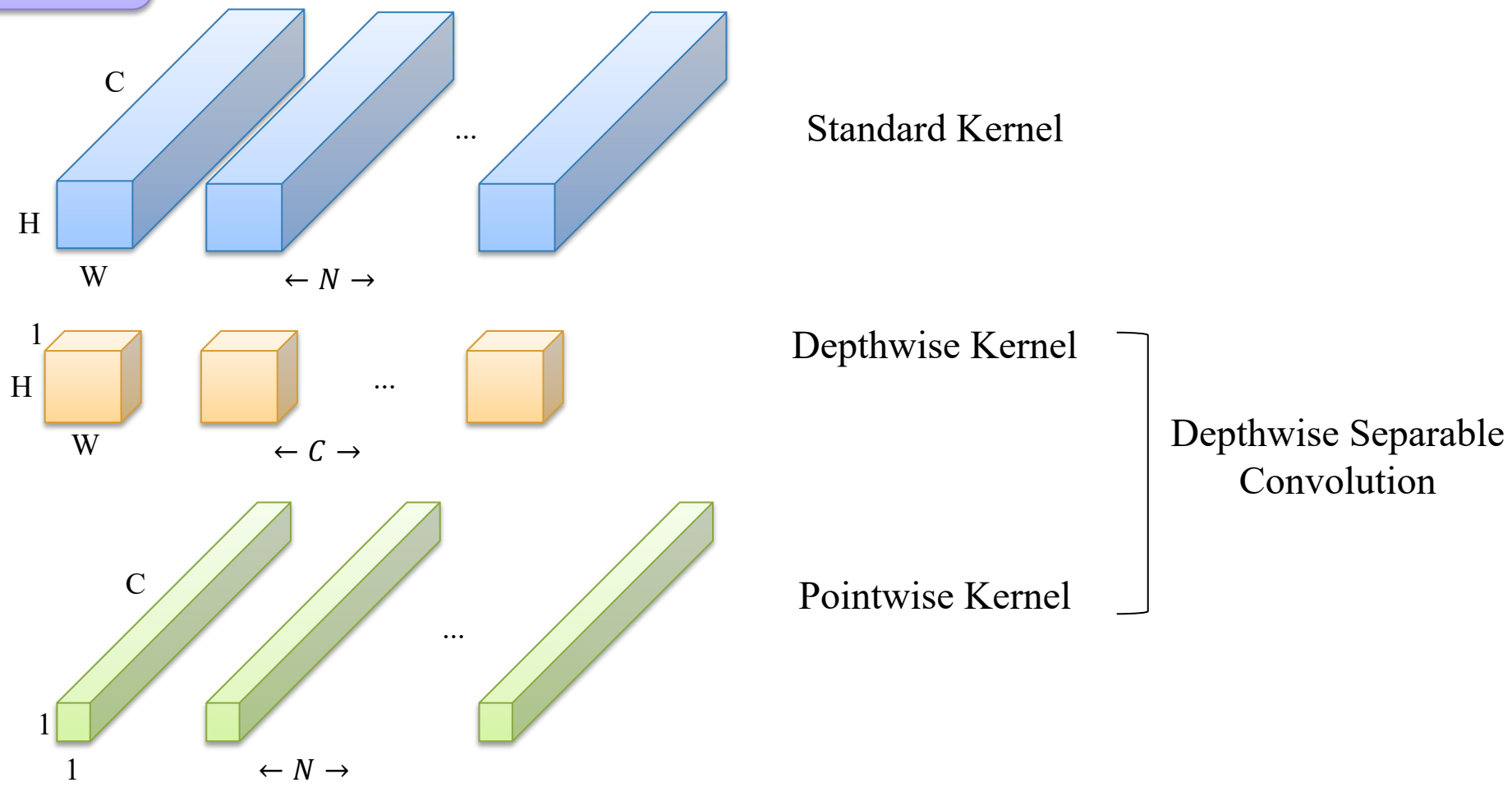
Use 3D Convs to explicitly extract features from the input video

Computional expensive



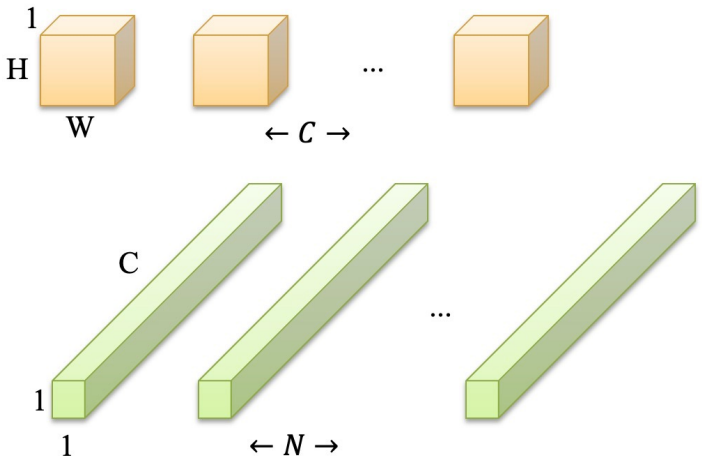
Video Classification: 3D CNN

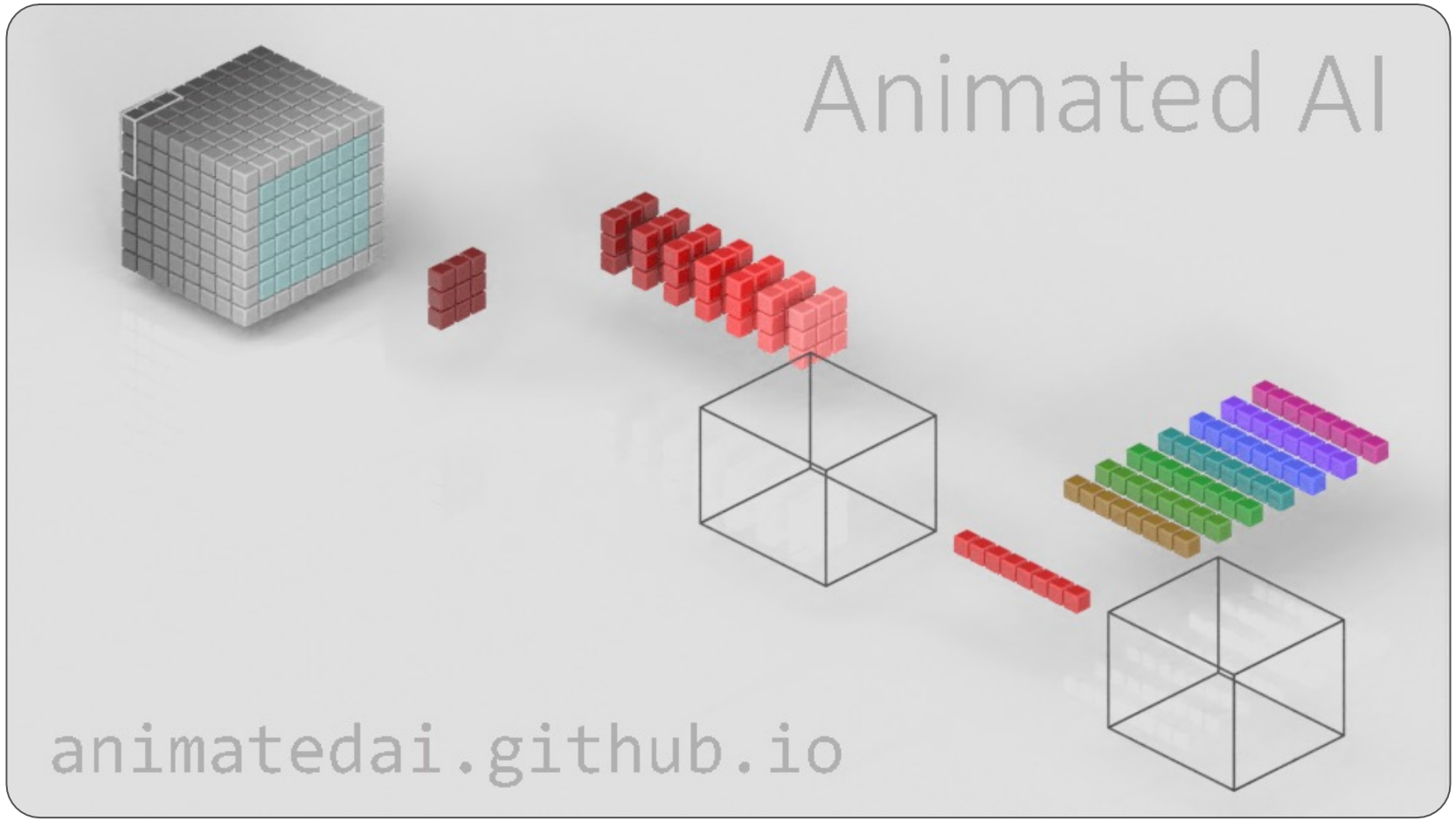
A step back: Depthwise Separable Convolution (in MobileNet)



Video Classification: 3D CNN

Depthwise Separable Convolution





https://animatedai.github.io/

AI VIET NAM
@aivietnam.edu.vn

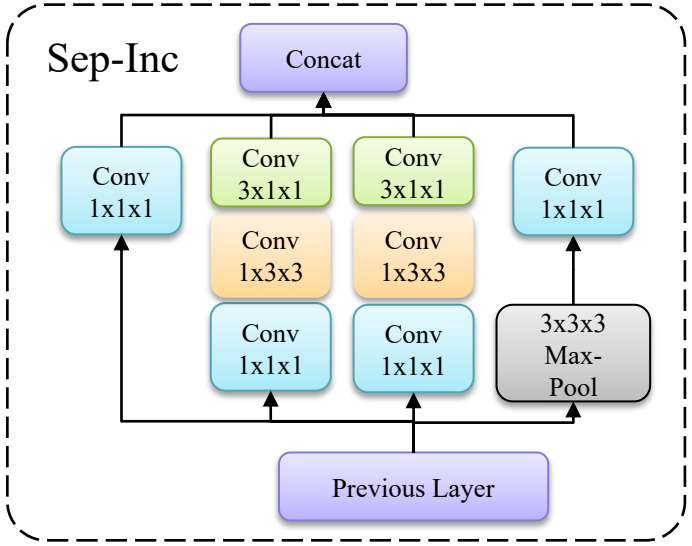
Video Classification: 3D CNN

S3D Model

Sep-Conv 7x7x7 Stride 2 → 1x3x3 Max-Pool Stride 1,2,2 → Conv 1x1x1 → Sep-Conv 3x3x3 → 1x3x3 Max-Pool Stride 1,2,2 → Sep-Inc → Sep-Inc → 3x3x3 Max-Pool Stride 2,2,2 → Sep-Inc → Sep-Inc → Sep-Inc → Sep-Inc → Sep-Inc → 3x3x3 Max-Pool Stride 2,2,2 → Sep-Inc → Sep-Inc → 3x3x3 Max-Pool Stride 2,2,2 → Conv 1x1x1 → prediction

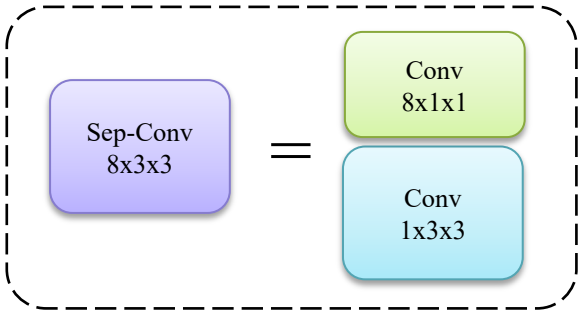
Sep-Conv 3x3x3 = Conv 3x1x1 → Conv 1x3x3

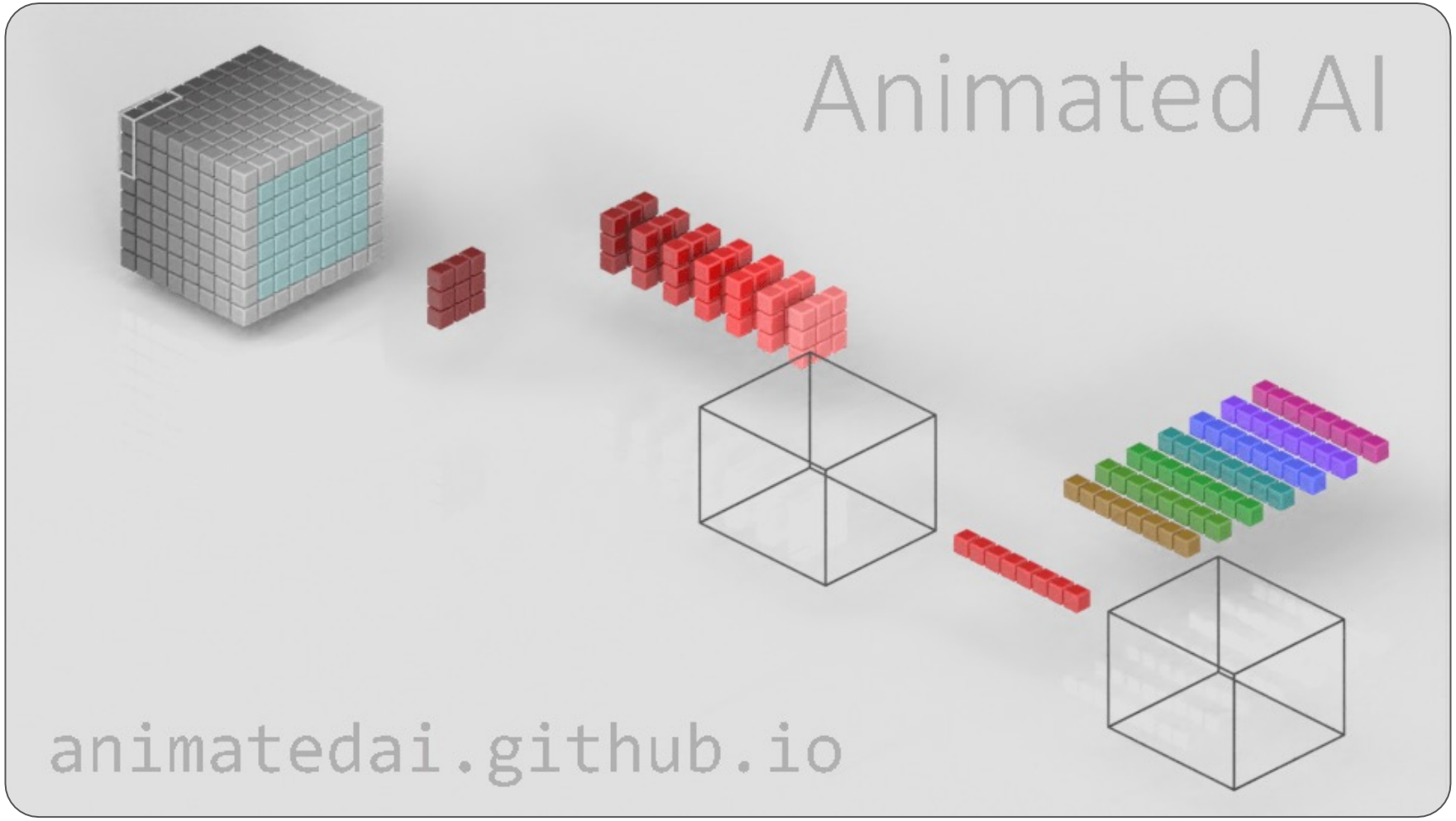
Sep-Inc



Video Classification: 3D CNN

Sep-Conv block in S3D





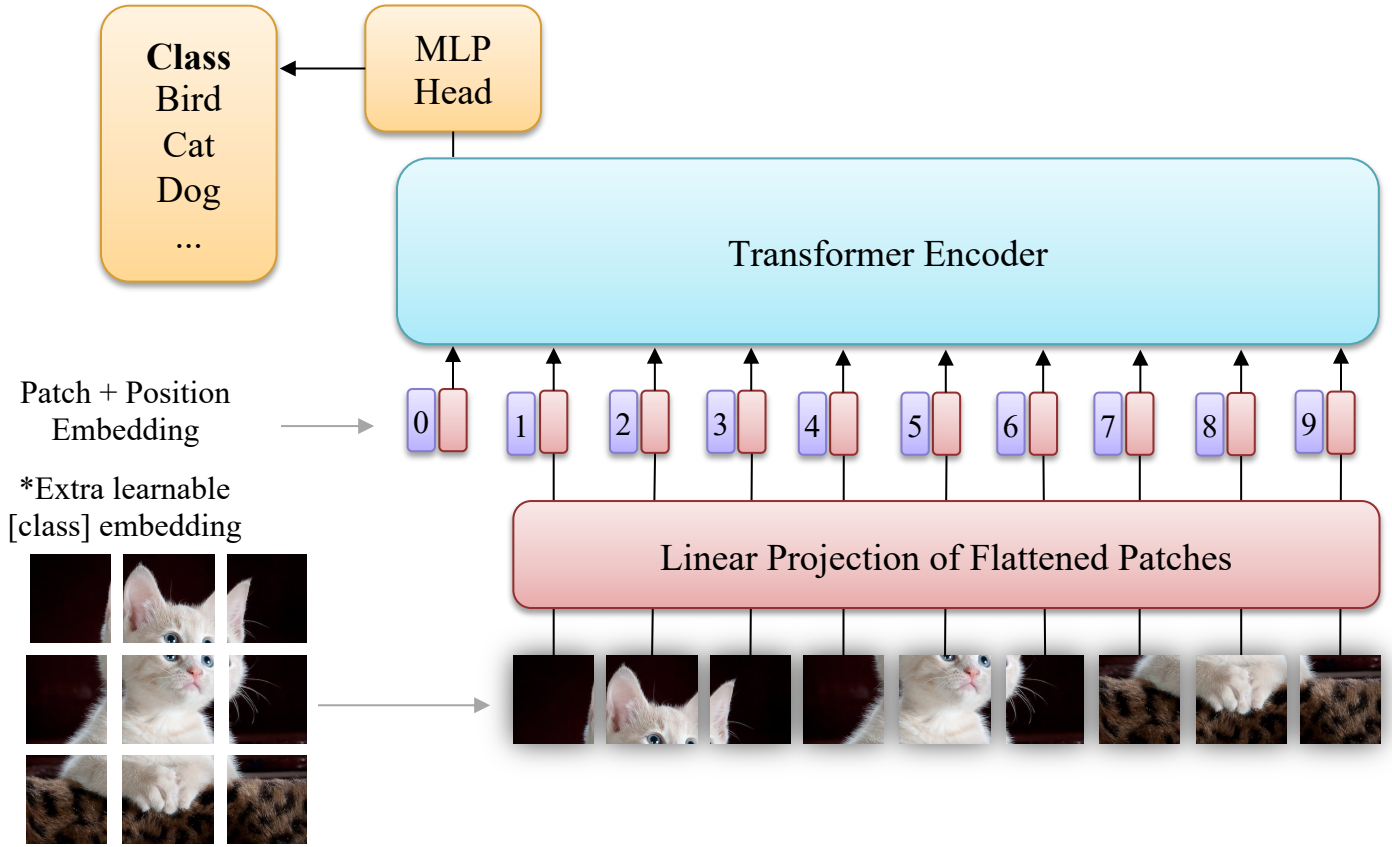
https://animatedai.github.io/

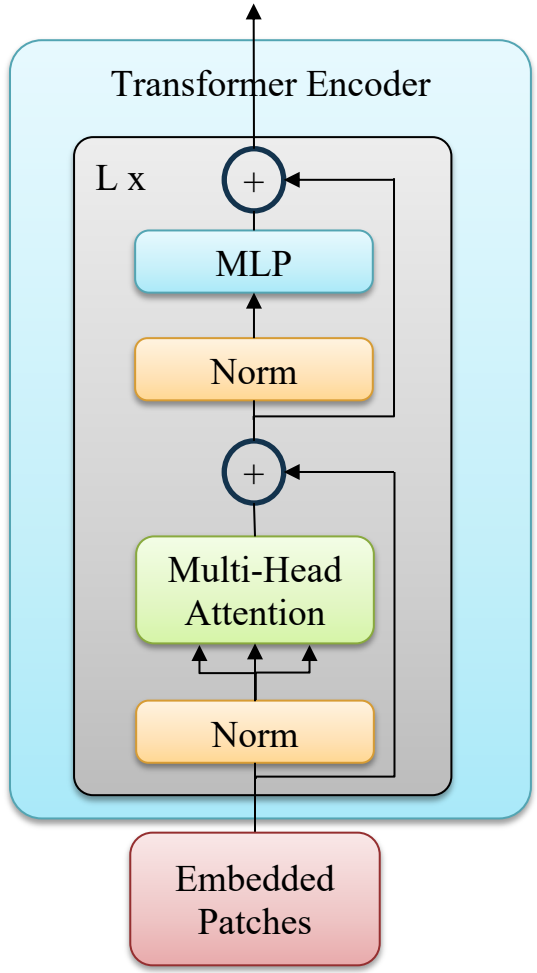
Video Classification: S3D

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3
CNN-LSTM	15.3M	63.0
CNN-Mamba	16.4M	83.0
3D CNN	9.1M	78.8

Video Classification: Video ViT (ViViT)

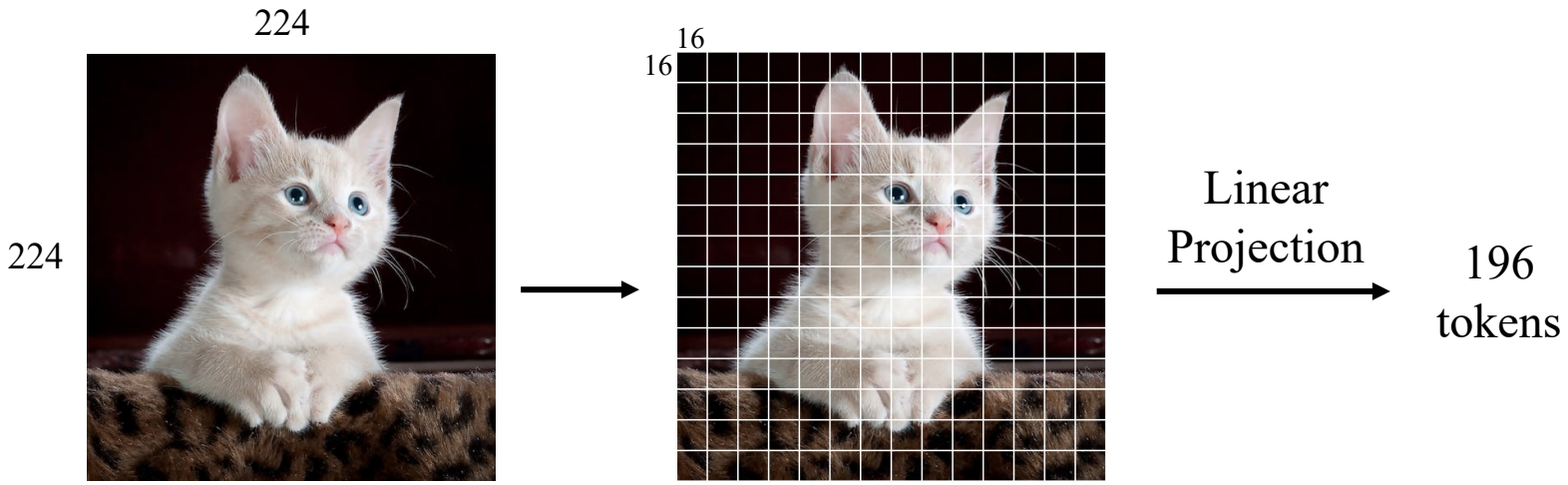
A step back: Vision Transformer (ViT)





Video Classification: Video ViT (ViViT)

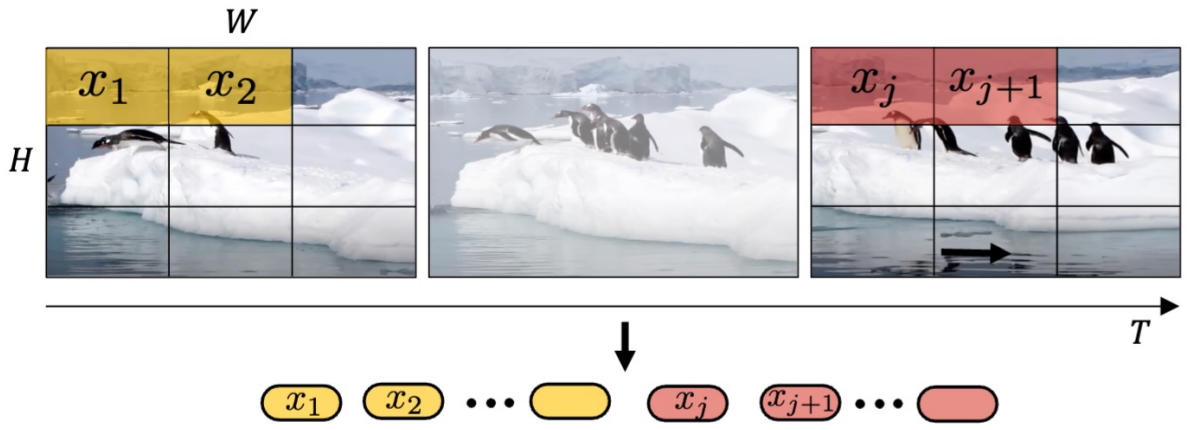
What are tokens in **images**?



Video Classification: Video ViT (ViViT)

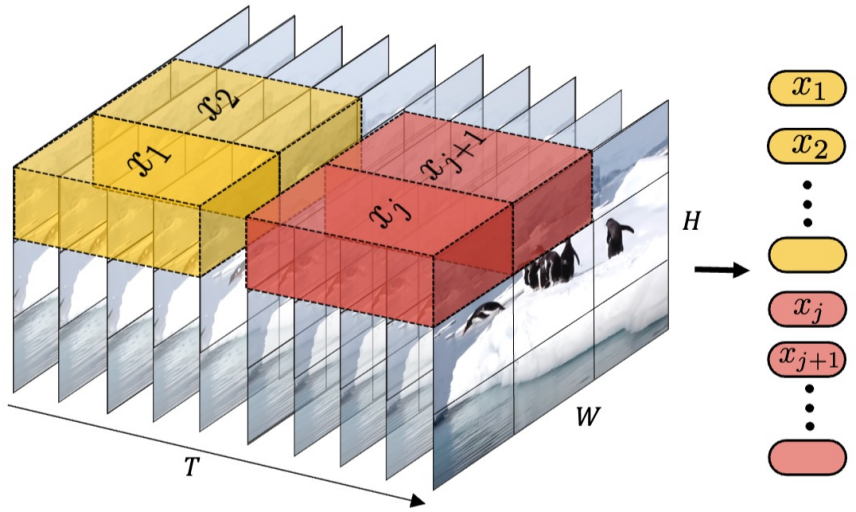
What are tokens in **videos**?

Uniform frame sampling



- uniformly sample n_t frames from the input
- embed each 2D frame independently using the same method as ViT
- concatenate all these tokens together

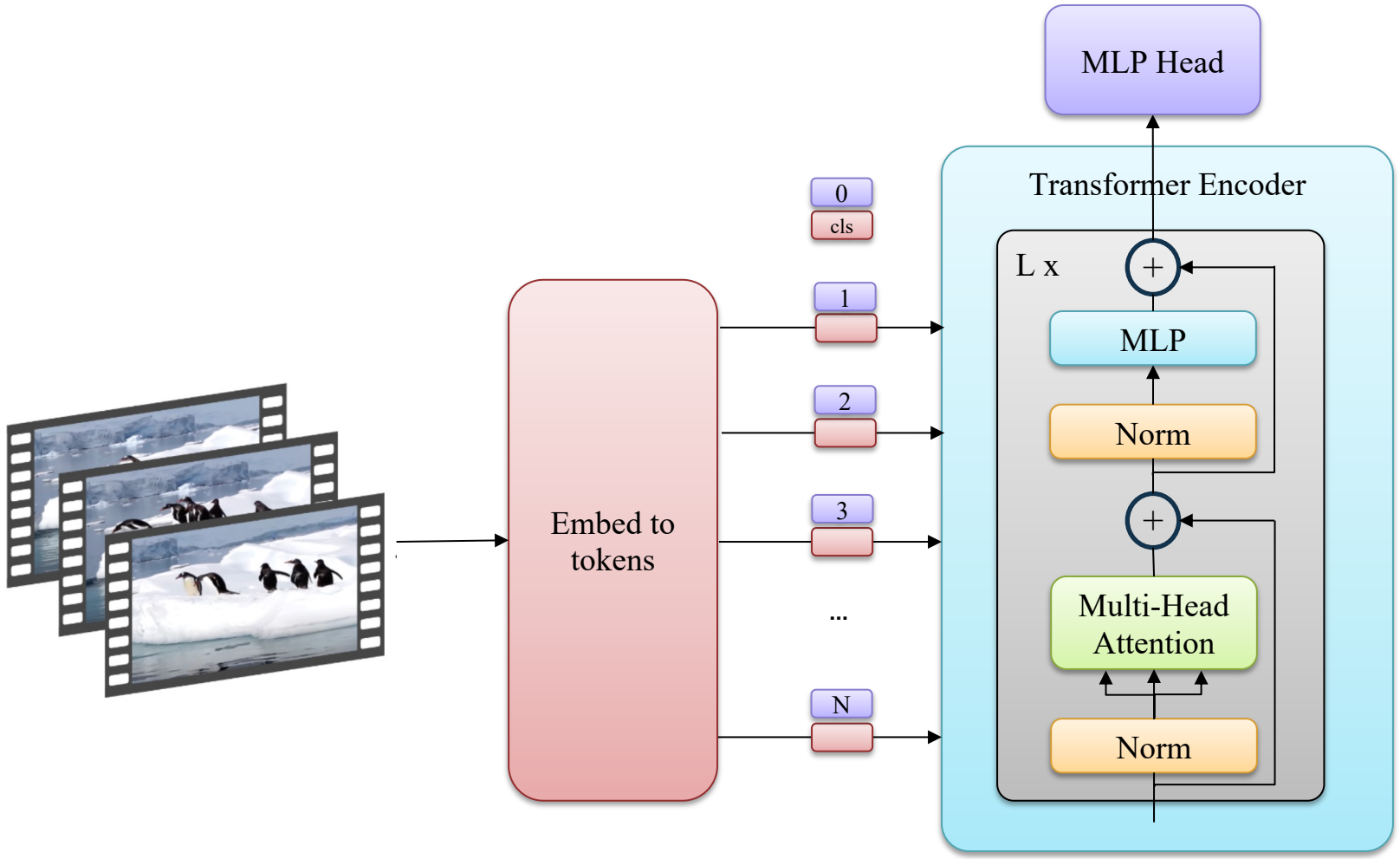
Tubelet embedding



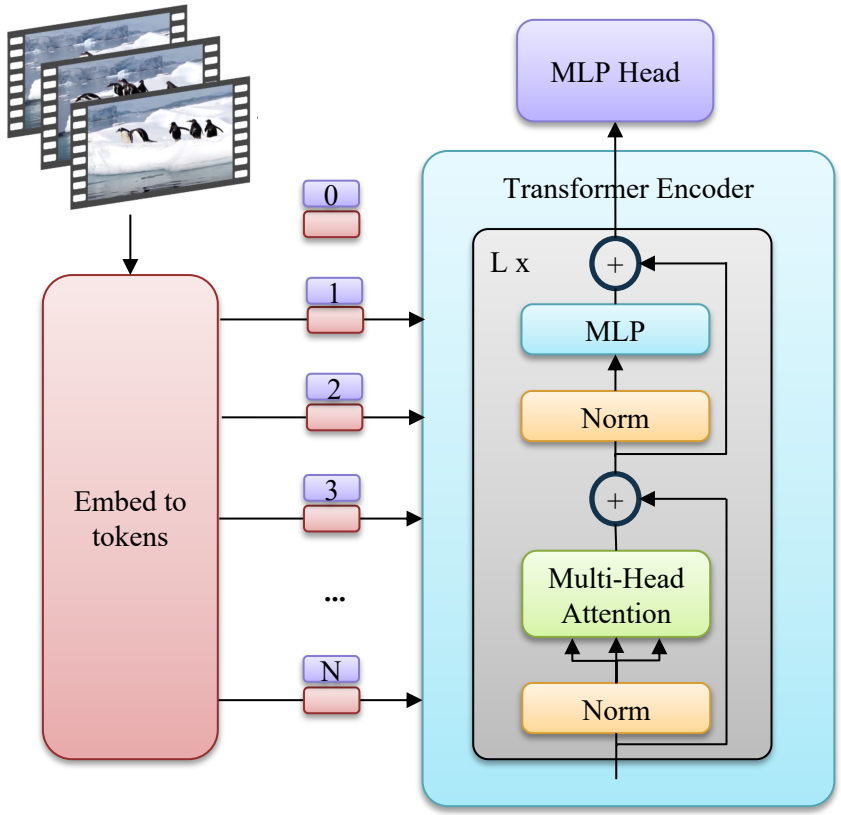
- fuse spatio-temporal information during tokenisation
- extract non-overlapping, spatio-temporal “tubes” from the input volume

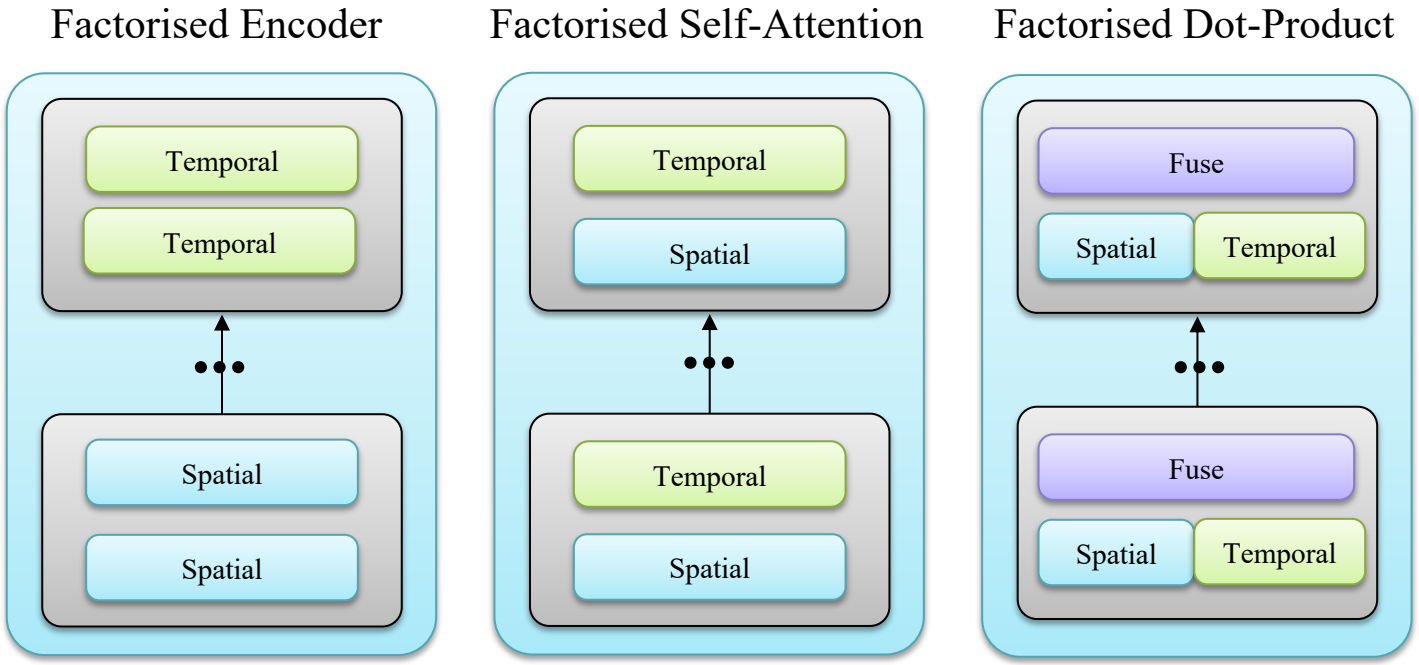
Arnab, Anurag, et al. "Vivit: A video vision transformer." *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.

Video Classification: Video ViT (ViViT)



Video Classification: Video ViT (ViViT)



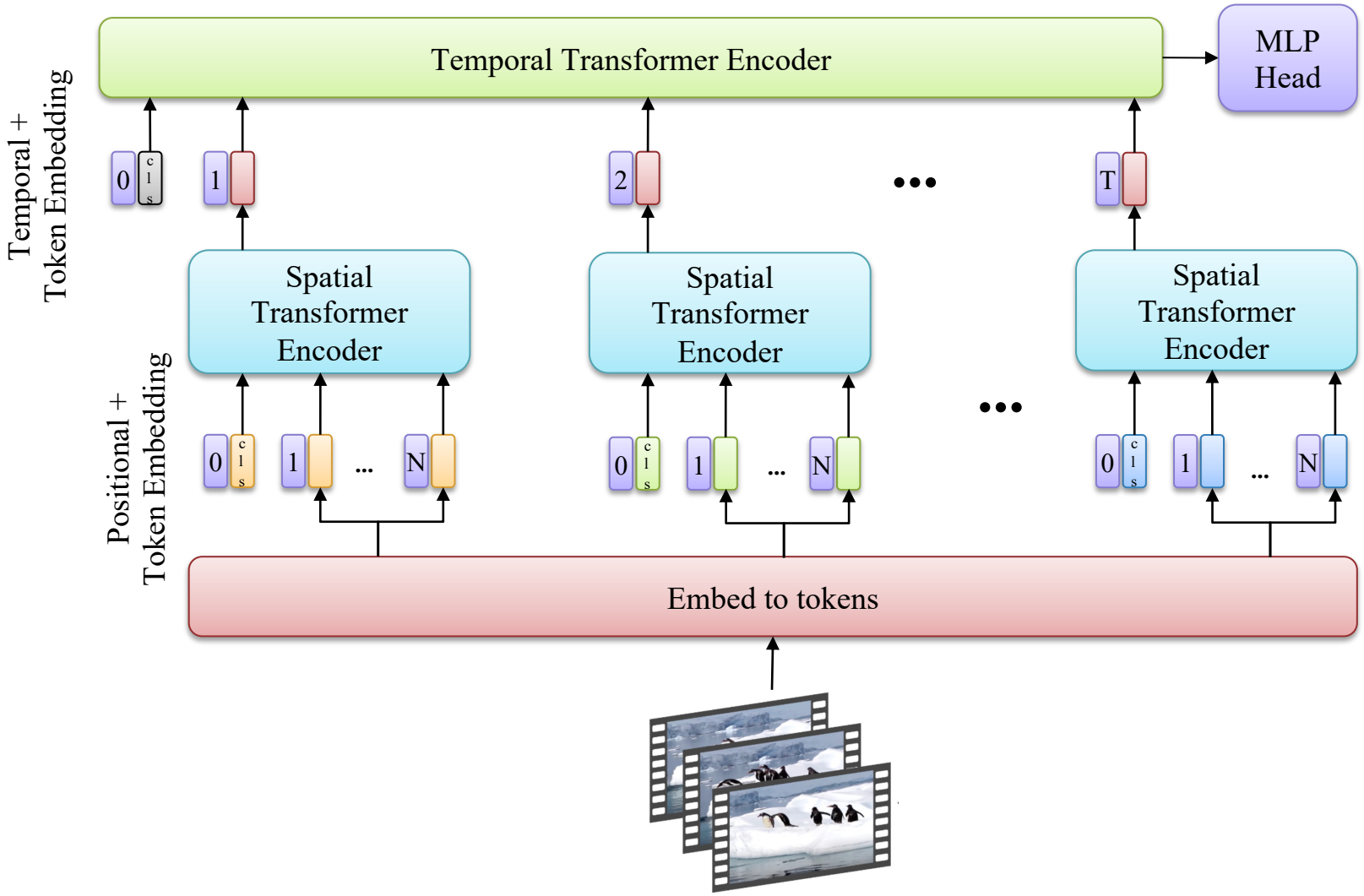


	K400	EK	FLOPs ($\times 10^9$)	Params ($\times 10^6$)	Runtime (ms)
Model 1: Spatio-temporal	80.0	43.1	455.2	88.9	58.9
Model 2: Fact. encoder	78.8	43.7	284.4	115.1	17.4
Model 3: Fact. self-attention	77.4	39.1	372.3	117.3	31.7
Model 4: Fact. dot product	76.3	39.5	277.1	88.9	22.9

Arnab, Anurag, et al. "Vivit: A video vision transformer." *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.

Video Classification: Video ViT (ViViT)

Factorised Encoder



Video Classification: Video ViT (ViViT)

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3
CNN-LSTM	15.3M	63.0
CNN-Mamba	16.4M	83.0
3D CNN	9.1M	78.8
Video ViT (ViViT)	86.5M	**84.8**

AI VIET NAM
@aivietnam.edu.vn

AI VIET NAM
@aivietnam.edu.vn

Quiz Time!!!

1. Đâu là một cách xử lý video có kích thước lớn?

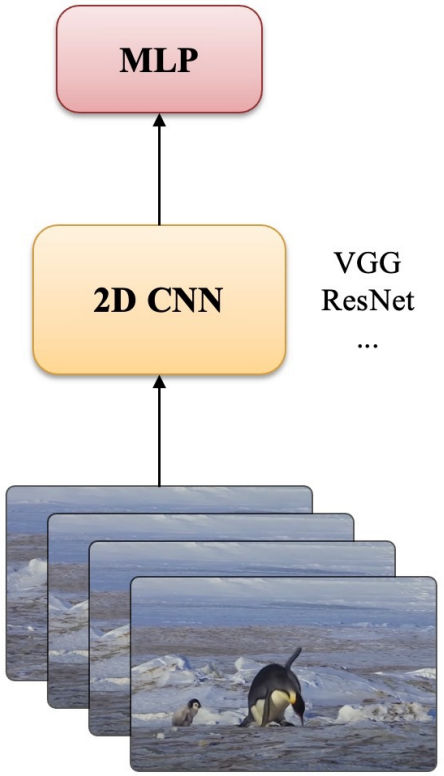
A. Sử dụng toàn bộ frame trong video.
B. Chia video thành nhiều clip ngắn và train model.
C. Bỏ qua các frame không quan trọng.
D. Sử dụng video với tốc độ khung hình cao.

2. Đâu là đặc điểm của data trong video classification?

A. Video là chuỗi của các frame theo không gian.
B. Mỗi frame thường có kích thước nhỏ.
C. Video là chuỗi của các frame theo thời gian.
D. Video không chứa dữ liệu âm thanh.

3. Trong early fusion, các frame được kết hợp như thế nào trước khi đưa vào model?

A. Tất cả các frames được nén lại thành một frame.
B. Các frames được giữ nguyên và xử lý độc lập.
C. Các frames được đưa vào LSTM như là các input độc lập.
D. Các frame được kết hợp để tạo thành tensor có kích thước (3*T x H x W).



AI VIET NAM
@aivietnam.edu.vn

Quiz Time!!!

4. Single-frame model hoạt động dựa trên nguyên tắc nào?

A. Tổng hợp các feature vector từ mỗi frame.
B. Dùng 2D model để predict trên mỗi frame và tổng hợp kết quả.
C. Xử lý từng frame với một mạng LSTM.
D. Kết hợp tất cả frames thành một tensor 3D.

5. Trong VideoDataset, hàm `uniform_sample` dùng để làm gì?

A. Để sắp xếp các frames theo thứ tự số.
B. Để chuyển đổi các frame sang RGB.
C. Để lấy mẫu đều các frame từ danh sách.
D. Để kết nối với GPU cho việc training nhanh hơn.

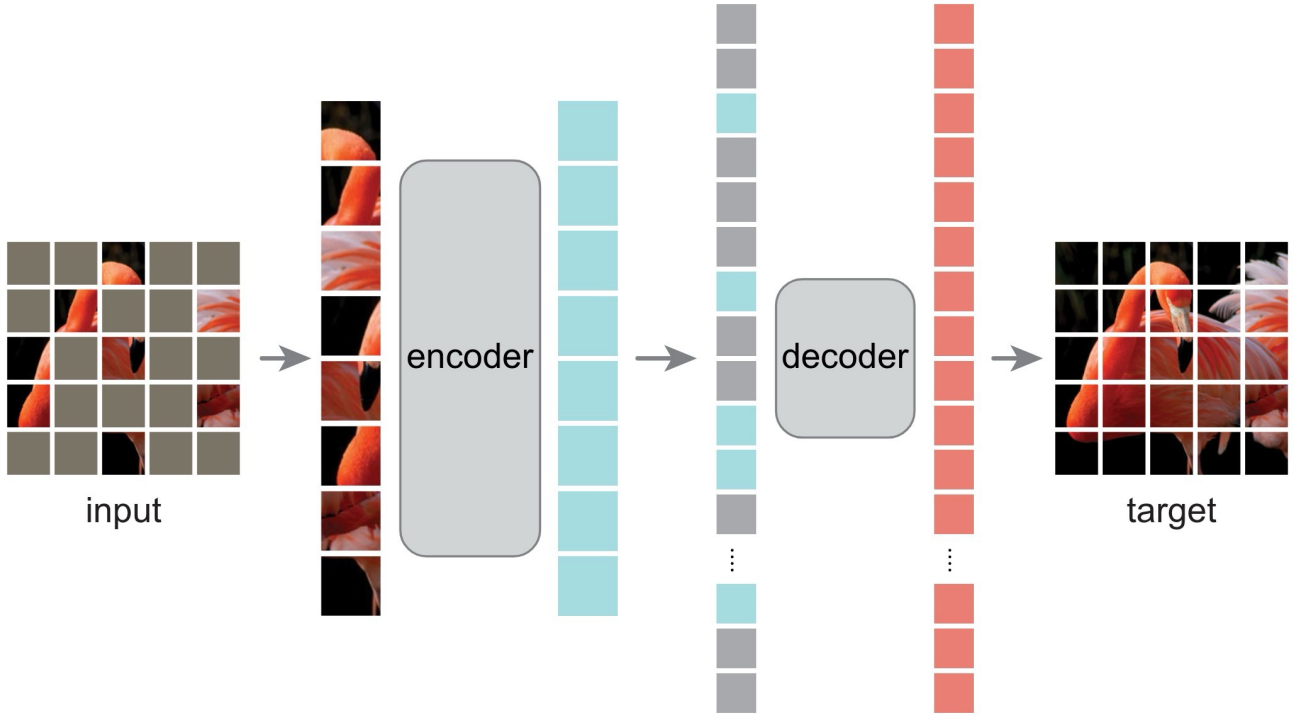
```python
def _uniform_sample(self, frames, n_frames):
    """
    Helper method to uniformly sample n_frames from the frames list.
    """
    stride = max(1, len(frames) // n_frames)
    sampled = [frames[i] for i in range(0, len(frames), stride)]
    return sampled[:n_frames]
```

Outline

1. Overview of Video Data
2. RWF2000 Dataset for Violence Detection Task
3. Models for Video Classification Task
4. Optional: Self-Supervised Learning for Video Data

Optional
Self-Supervised Learning: VideoMAE

A step back: Mask Autoencoder (MAE)

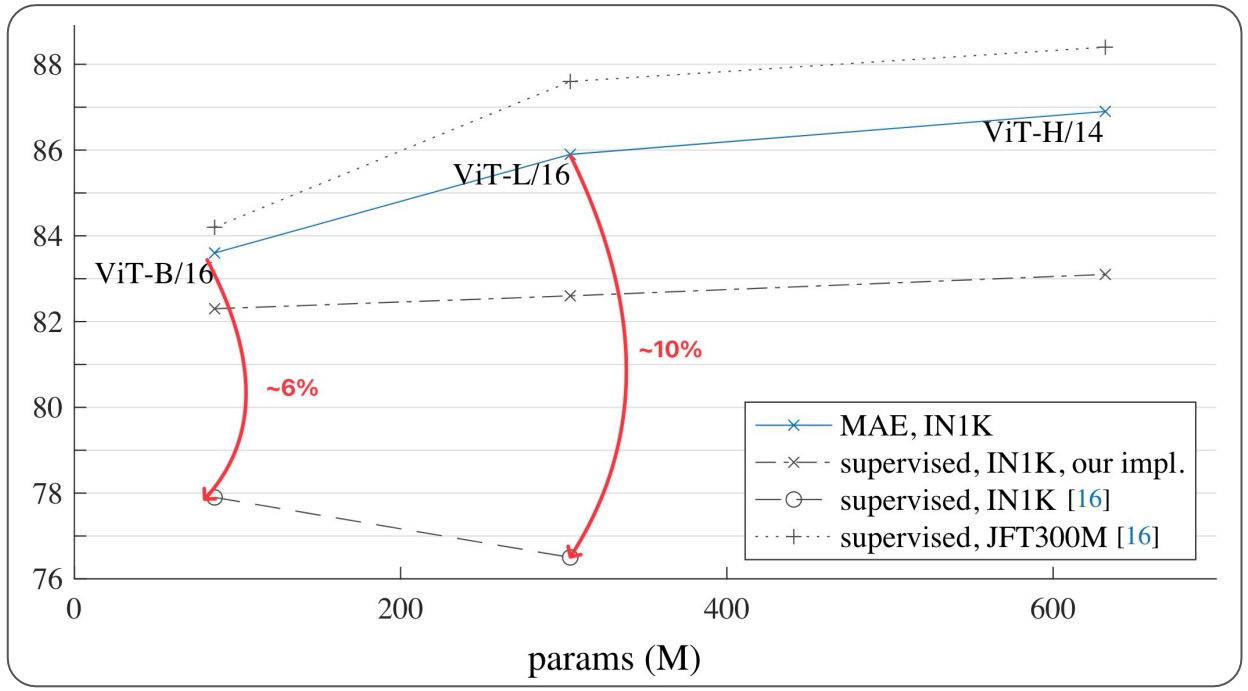


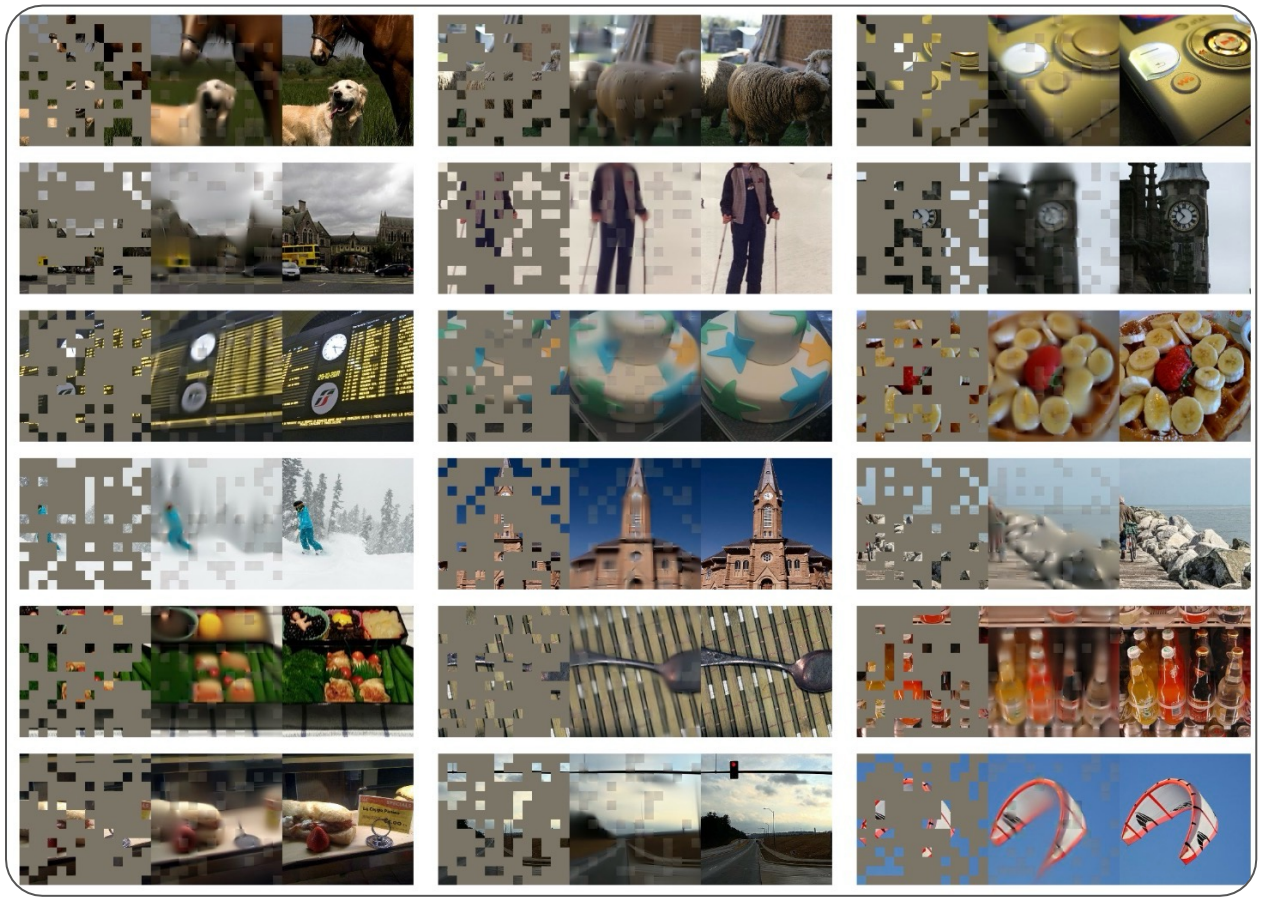
He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2021). Masked Autoencoders Are Scalable Vision Learners. ArXiv. /abs/2111.06377

AI VIET NAM
@aivietnam.edu.vn

Optional
Self-Supervised Learning: VideoMAE

A step back: Mask Autoencoder (MAE)

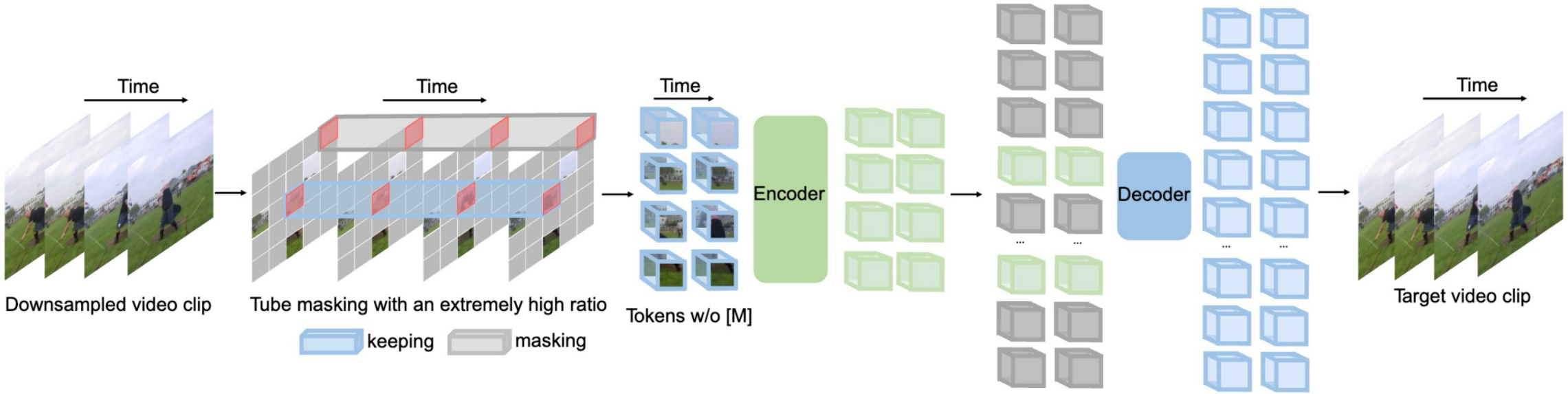




He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2021). Masked Autoencoders Are Scalable Vision Learners. ArXiv. /abs/2111.06377

Optional
Self-Supervised Learning: VideoMAE

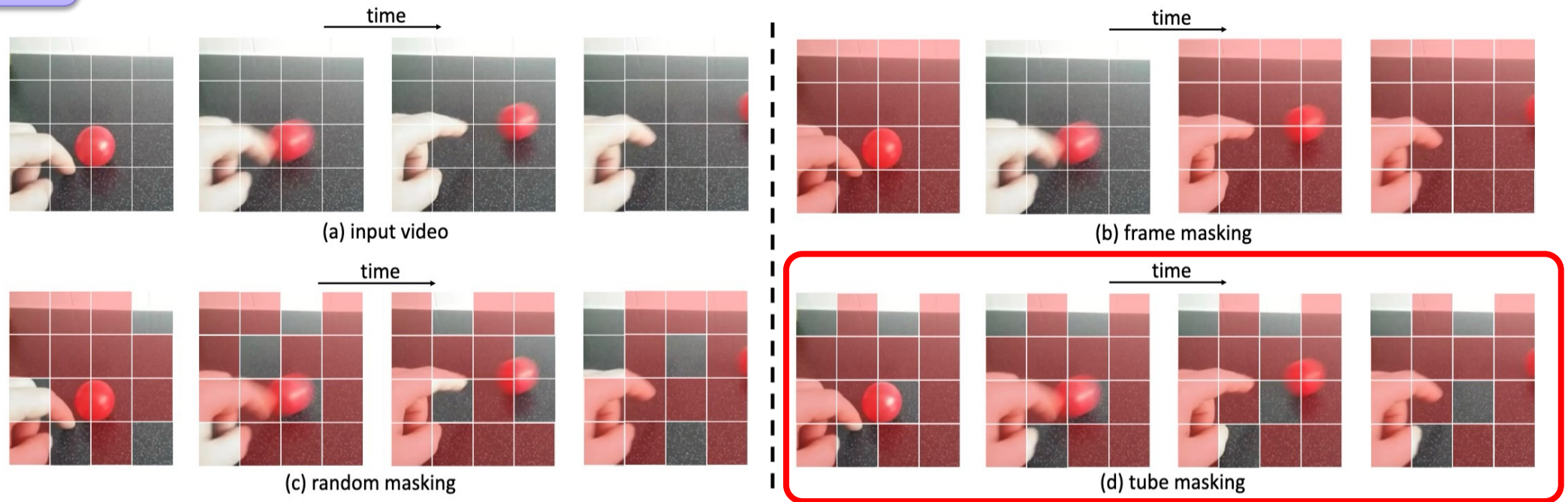
VideoMAE



Tong, Z., Song, Y., Wang, J., & Wang, L. (2022). VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. ArXiv. /abs/2203.12602

Optional
Self-Supervised Learning: VideoMAE

VideoMAE



case	ratio	SSV2	K400
tube	75	68.0	79.8
tube	90	**69.6**	**80.0**
random	90	68.3	79.5
frame	87.5*	61.5	76.5

Reduce information leakage

Tong, Z., Song, Y., Wang, J., & Wang, L. (2022). VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. ArXiv. /abs/2203.12602

Optional
Self-Supervised Learning: VideoMAE

Comparison on **Something-Something** dataset

Method	Backbone	Extra data	Ex. labels	Frames	GFLOPs	Param	Top-1	Top-5
TEINet$_{En}$ [40]	ResNet50$_{\times 2}$	ImageNet-1K	✓	8+16	99×10×3	50	66.5	N/A
TANet$_{En}$ [41]	ResNet50$_{\times 2}$		✓	8+16	99×2×3	51	66.0	90.1
TDN$_{En}$ [75]	ResNet101$_{\times 2}$		✓	8+16	198×1×3	88	69.6	92.2
SlowFast [23]	ResNet101	Kinetics-400	✓	8+32	106×1×3	53	63.1	87.6
MViTv1 [22]	MViTv1-B		✓	64	455×1×3	37	67.7	90.9
TimeSformer [6]	ViT-B	ImageNet-21K	✓	8	196×1×3	121	59.5	N/A
TimeSformer [6]	ViT-L		✓	64	5549×1×3	430	62.4	N/A
ViViT FE [3]	ViT-L	IN-21K+K400	✓	32	995×4×3	N/A	65.9	89.9
Motionformer [51]	ViT-B		✓	16	370×1×3	109	66.5	90.1
Motionformer [51]	ViT-L		✓	32	1185×1×3	382	68.1	91.2
Video Swin [39]	Swin-B		✓	32	321×1×3	88	69.6	92.7
VIMPAC [65]	ViT-L	HowTo100M+DALLE	✗	10	N/A×10×3	307	68.1	N/A
BEVT [77]	Swin-B	IN-1K+K400+DALLE	✗	32	321×1×3	88	70.6	N/A
MaskFeat↑312 [80]	MViT-L	Kinetics-600	✓	40	2828×1×3	218	75.0	95.0
VideoMAE	ViT-B	Kinetics-400	✗	16	180×2×3	87	69.7	92.3
VideoMAE	ViT-L	Kinetics-400	✗	16	597×2×3	305	74.0	94.6
VideoMAE	ViT-S	*no external data*	✗	16	57×2×3	22	66.8	90.3
VideoMAE	ViT-B		✗	16	180×2×3	87	70.8	92.4
VideoMAE	ViT-L		✗	16	597×2×3	305	74.3	94.6
VideoMAE	ViT-L		✗	32	1436×1×3	305	**75.4**	**95.2**

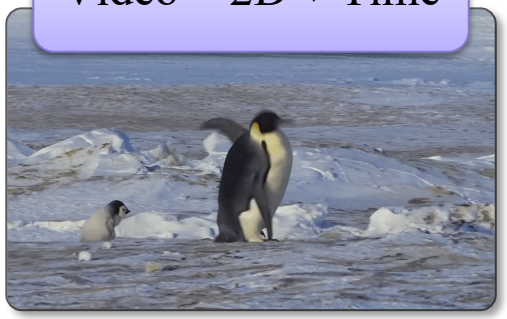
Tong, Z., Song, Y., Wang, J., & Wang, L. (2022). VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. ArXiv. /abs/2203.12602

Video Classification: Video ViT (ViViT)

	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3
CNN-LSTM	15.3M	63.0
CNN-Mamba	16.4M	83.0
3D CNN	9.1M	78.8
Video ViT (ViViT)	86.5M	<u>84.8</u>
VideoMAE	86.2M	**91.3**

Summary

Video = 2D + Time

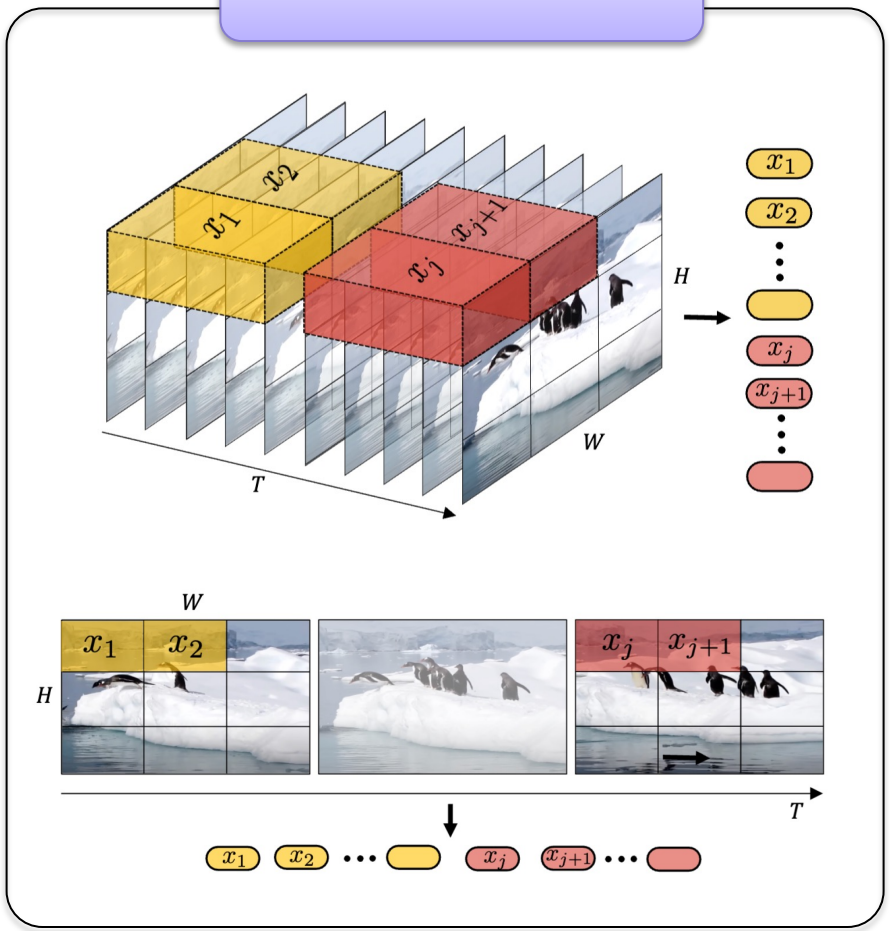


T x 3 x H x W

Videos are big!

Solution: train on short **clips**: low fps and low spatial resolution

Tokens in Video Data



	#params	Best Val Accuracy
Single-Frame CNN	11.5M	72.8
Late Fusion	11.5M	73.0
Early Fusion	11.6M	76.3
CNN-LSTM	15.3M	63.0
CNN-Mamba	16.4M	83.0
3D CNN	9.1M	78.8
Video ViT (ViViT)	86.5M	84.8
VideoMAE	86.2M	**91.3**

AI VIET NAM

@aivietnam.edu.vn

Thanks!

Any questions?