Computer Vision - 2025

Week #11. Video Data Processing

Lectures by Alexei Kornaev [1,2,3]
Practical sessions by Kirill Yakovlev [2]

[1] AI Institute, Innopolis University (IU), Innopolis
[2] Robotics & CV Master's Program, IU, Innopolis
[3] Dept. of M^2R, Orel State University, Orel

[4] RC for AI, National RC for Oncology, Moscow

March 31, 2025

Agenda

1. Outcomes
2. Theoretical Background and Historical Context
3. Video Processing with ML
4. Conclusion

Section 1. Outcomes

Outcomes

This week's lecture and practical session on Video Data Processing aim to provide an understanding of motion analysis, temporal modeling, and deep learning-based video processing. By the end of this week, students will be able to:

1. Understand the principles of optical flow, video stabilization, action recognition, and video summarization.
2. Describe different temporal modeling approaches, including 2D CNNs + RNNs (LSTMs), 3D CNNs, and Video Transformers (ViViT).
3. Implement a video-based classification/approximation model using 2D CNNs, 2D CNN + LSTM, 3D CNNs, and ViViT.

Key Takeaway: Video data processing is essential for applications in robotics, surveillance, autonomous driving, and human activity recognition, with deep learning playing a crucial role in modern advancements.

Section 2. Theoretical Background and Historical Context

Optical Flow Fundamentals

Definition & Importance

- **Optical Flow:** The apparent motion of brightness patterns in a sequence of images.
- Provides information about object movement and scene structure.
- Critical for tasks such as motion estimation, tracking, and video stabilization.

Mathematical Formulation

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0, \tag{1}$$

where u and v denote the horizontal and vertical flow components.

Historical Context

- Early work by Horn & Schunck Horn and Schunck [1981] laid the foundation.
- Nowadays the optical flow methods are implemented with DL models, e.g. RAFT.

Video Stabilization Techniques

Objective

- Remove unwanted camera motion (e.g., shake) to produce smooth videos.
- Enhance the visual quality and reliability for further processing.

Methodologies

- Feature tracking and estimation of global motion.
- Optimization techniques using robust norms (e.g., L1 minimization).

Notable Contribution

- Auto-directed video stabilization with robust L1 optimal camera paths **?**.

Action Recognition & Video Summarization

Action Recognition

- Transition from handcrafted features to deep learning approaches.
- Two-stream CNNs capture spatial and temporal information **?**.
- 3D CNNs extend the concept to spatio-temporal feature extraction **?**.

Video Summarization

- Identifying keyframes and key events from long video sequences.
- Balancing unsupervised vs. supervised approaches **?**.

Method	**Temporal Modeling**	**Reference**
Two-Stream CNN	Frame-wise + Temporal Fusion	**?**
3D CNN	Direct Spatio-temporal Extraction	**?**
Summarization	Keyframe/Event Extraction	**?**

Table: Comparative Overview of Methods

Section 3. Video Processing with ML

Recap: 2D Convolution Applied to an Image

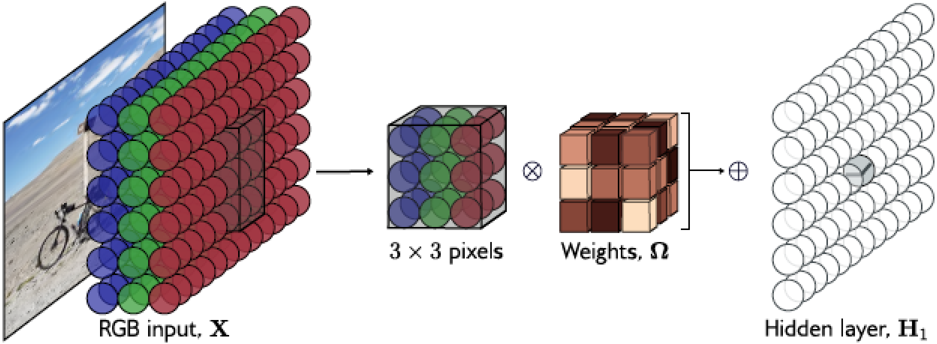


Figure: The image is treated as a 2D input with three channels corresponding to the red, green, and blue components. With a 3×3 kernel, each pre-activation in the first hidden layer is computed by point-wise multiplying the $3 \times 3 \times 3$ kernel weights with the 3×3 RGB image patch centered at the same position, summing, and adding the bias. To calculate all the pre-activations in the hidden layer, we "slide" the kernel over the image in both horizontal and vertical directions. The output is a 2D layer of hidden units. To create multiple output channels, we would repeat this process with multiple kernels, resulting in a 3D tensor of hidden units at hidden layer H_1 [Prince, 2023].

CV-2025

A.Kornaev,
K.Yakovlev

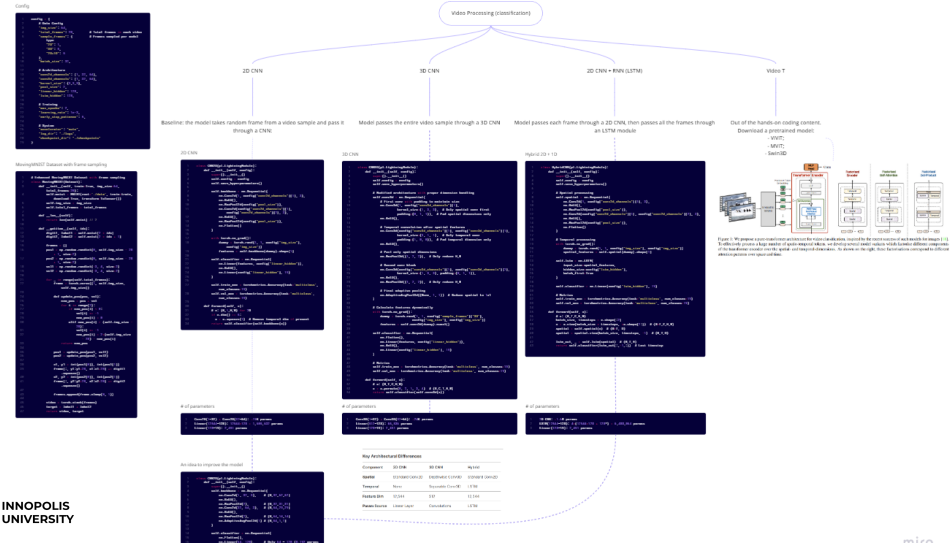
Outcomes

Theoretical Background and Historical Context

Video Processing with ML

Conclusion

Approaches to a video classification task



Config

```
config = {
    # Data Config
    "img_size": 64,
    "total_frames": 20,          # Total frames in each video
    "sample_frames": {           # Frames sampled per model
        type
        "2D": 1,
        "3D": 8,
        "2Dx1D": 8
    },
    "batch_size": 32,

    # Architecture
    "conv2d_channels": [1, 32, 64],
    "conv3d_channels": [1, 32, 64],
    "kernel_size": (3,3,3),
    "pool_size": 2,
    "linear_hidden": 128,
    "lstm_hidden": 128,

    # Training
    "max_epochs": 2,
    "learning_rate": 1e-3,
    "early_stop_patience": 5,

    # System
    "accelerator": "auto",
    "log_dir": "./logs",
    "checkpoint_dir": "./checkpoints"
}
```

MovingMNIST Dataset with frame sampling

```
# Enhanced MovingMNIST Dataset with frame sampling
class MovingMNIST(Dataset):
[illegible]
```

Video Processing (classification)

2D CNN

Baseline: the model takes random frame from a video sample and pass it through a CNN:

2D CNN

```
class CNN2D(pl.LightningModule):
    def __init__(self, config):
        super().__init__()
        self.config = config
        self.save_hyperparameters()

        self.backbone = nn.Sequential(
[illegible]
```

# of parameters

```
Conv2D(1→32) + Conv2D(32→64): [illegible] params
Linear([illegible]→128): [illegible] params
Linear(128→10): [illegible] params
```

An idea to improve the model

```
class CNN2D(pl.LightningModule):
    def __init__(self, config):
        super().__init__()
        self.backbone = nn.Sequential(
[illegible]
```

3D CNN

Model passes the entire video sample through a 3D CNN

3D CNN

```
class CNN3D(pl.LightningModule):
    def __init__(self, config):
        super().__init__()
        self.config = config
        self.save_hyperparameters()

        # Modified architecture with proper dimension handling
[illegible]
    def forward(self, x):
        # x: (B,T,C,H,W)
        x = x.permute(0, 2, 1, 3, 4)  # (B,C,T,H,W)
        return self.classifier(self.conv3d(x))
```

# of parameters

```
Conv3D(1→32) + Conv3D(32→64): [illegible] params
Linear(512→128): 65,536 params
Linear(128→10): [illegible] params
```

Key Architectural Differences

Component	2D CNN	3D CNN	Hybrid
Spatial	Standard Conv2D	Depthwise Conv3D	Standard Conv2D
Temporal	None	Separable Conv3D	LSTM
Feature Dim	12,544	512	12,544
Param Source	Linear Layer	Convolutions	LSTM

2D CNN + RNN (LSTM)

Model passes each frame through a 2D CNN, then passes all the frames through an LSTM module

Hybrid 2D + 1D

```
class HybridCNN(pl.LightningModule):
    def __init__(self, config):
        super().__init__()
        self.config = config
        self.save_hyperparameters()

        # Spatial processing
[illegible]
```

# of parameters

```
2D CNN: [illegible] params
LSTM(12544→128): 4×(12544×128 + 128²) = [illegible] params
Linear(128→10): [illegible] params
```

Video T

Out of the hands-on coding content.
Download a pretrained model:
- ViViT;
- MViT;
- Swin3D

Figure 1: We propose a pure-transformer architecture for video classification, inspired by the recent success of such models for images [illegible]. To effectively process a large number of spatio-temporal tokens, we develop several model variants which factorise different components of the transformer encoder over the spatial- and temporal-dimensions. As shown on the right, these factorisations correspond to different attention patterns over space and time.

INNOPOLIS UNIVERSITY

Paper reading

ViViT: A Video Vision Transformer

We present pure-transformer based models for video classification, drawing upon the recent success of such models in image classification. Our model extracts spatio-temporal tokens from the input video, which are then encoded by a series of transformer layers. In order to handle the long sequences of tokens encountered in video, we propose several, efficient variants of our model which factorise the spatial- and temporal-dimensions of the input. Although transformer-based models are known to only be effective when large training datasets are available, we show how we can effectively regularise the model during training and leverage pretrained image models to be able to train on comparatively small datasets. We conduct thorough ablation studies, and achieve state-of-the-art results on multiple video classification benchmarks including Kinetics 400 and 600, Epic Kitchens, Something-Something v2 and Moments in Time, outperforming prior methods based on deep 3D convolutional networks. To facilitate further research, we will release code and models Arnab et al. [2021].

Section 4. Conclusion

Conclusion

In this lecture, we explored the fundamental concepts and modern advancements in video data processing. We covered:

- Optical Flow: Classical methods (Lucas-Kanade, Horn-Schunck) and deep learning-based models (FlowNet, PWC-Net, RAFT).
- Video Stabilization & Action Recognition: Key techniques for video preprocessing and activity classification.
- Temporal Modeling: Comparing RNNs, LSTMs, 3D CNNs, and Video Transformers.
- Hands-on Applications: Training models for classification.

Key Takeaway: Modern video processing leverages deep learning to improve motion estimation, activity recognition, and temporal modeling. Transformers are emerging as a powerful tool for video understanding.

Bibliography

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario LuÄiÄ, and Cordelia Schmid. Vivit: A video vision transformer, 2021. URL https://arxiv.org/abs/2103.15691.

Berthold KP Horn and Brian G Schunck. Determining optical flow. **Artificial Intelligence**, 17(1-3):185–203, 1981.

Simon J.D. Prince. **Understanding Deep Learning**. The MIT Press, 2023. URL http://udlbook.com.