
Politecnico di Torino

AI ML

# Exercises

## Advanced Machine Learning

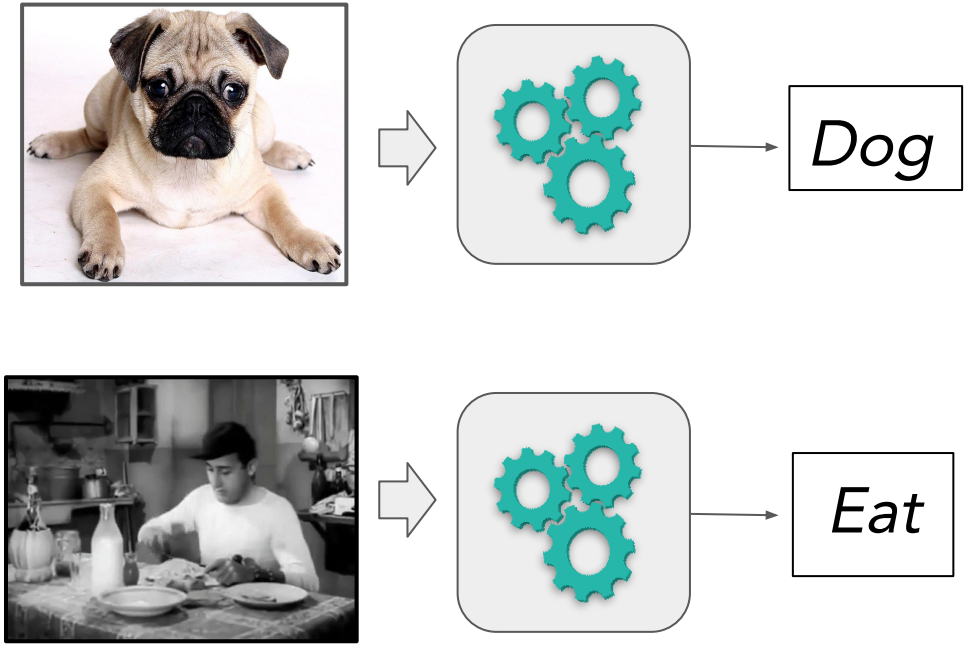
Teaching Assistant:
**Paolo Rabino**
paolo.rabino@polito.it

# Overview

1. Train a **C**onvolutional **LSTM** for First Person Action Recognition (FPAR):
    - Network: **ResNet 34 + RNN**
    - Videos: **GTEA61**

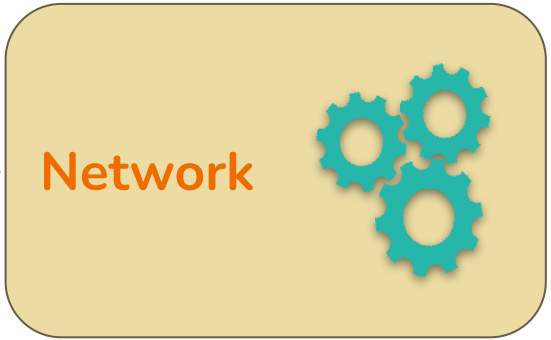
2. Exercise **Steps**:
    - Learning without Temporal Information (Avgpool)
    - Learning with Temporal Information (LSTM)
    - Learning with Spatio-Temporal Information (ConvLSTM)

# Introduction



# Introduction

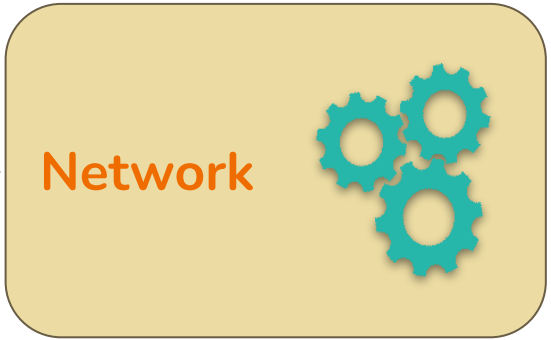
# Introduction

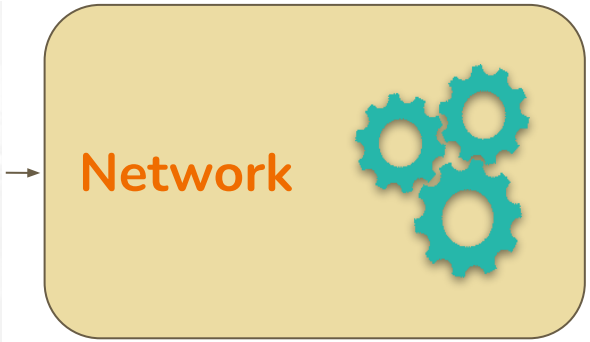
# Introduction

# Introduction

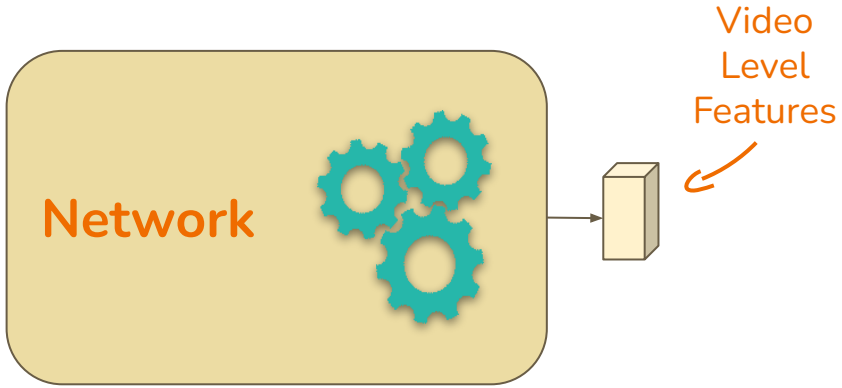
# Introduction

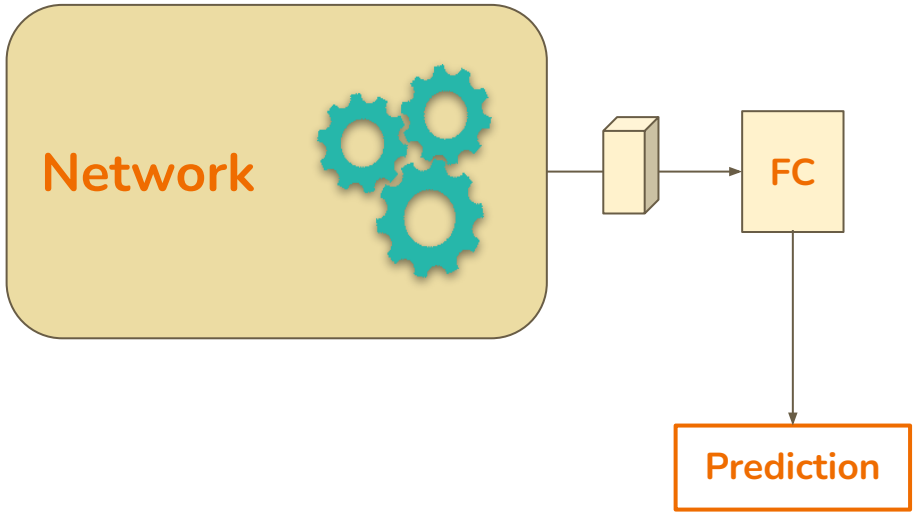
# Introduction

# Introduction

# Introduction

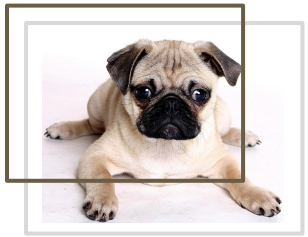
# Image vs Frames

# Image vs Frames

Don’t worry! Already Implemented!!!

## Training

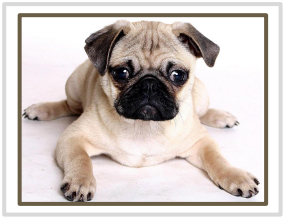
Random Crop



Random Crop + **Random Sampling**

## Test

Center Crop

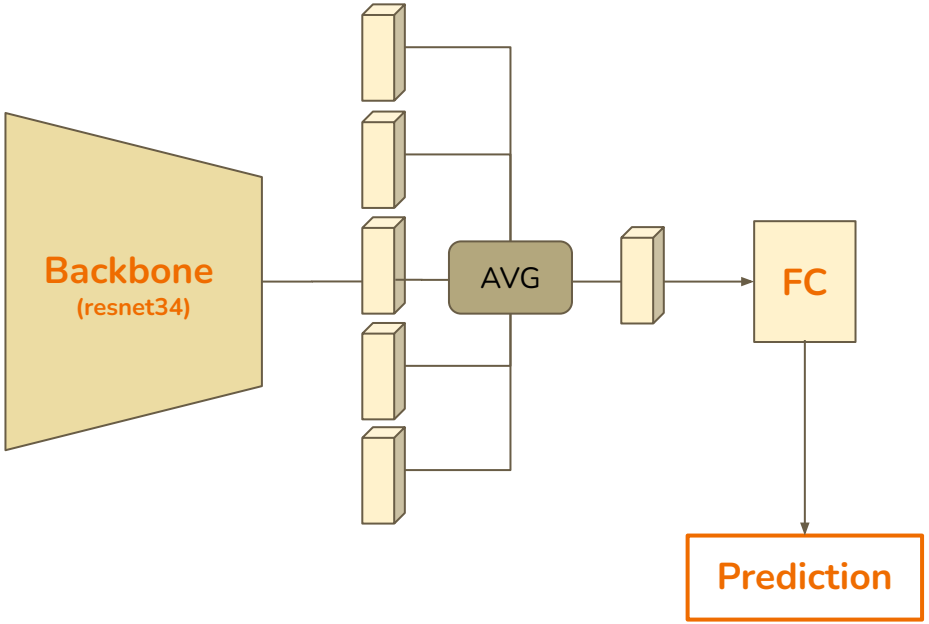


Center Crop + **Uniform Sampling**

# Network - 0

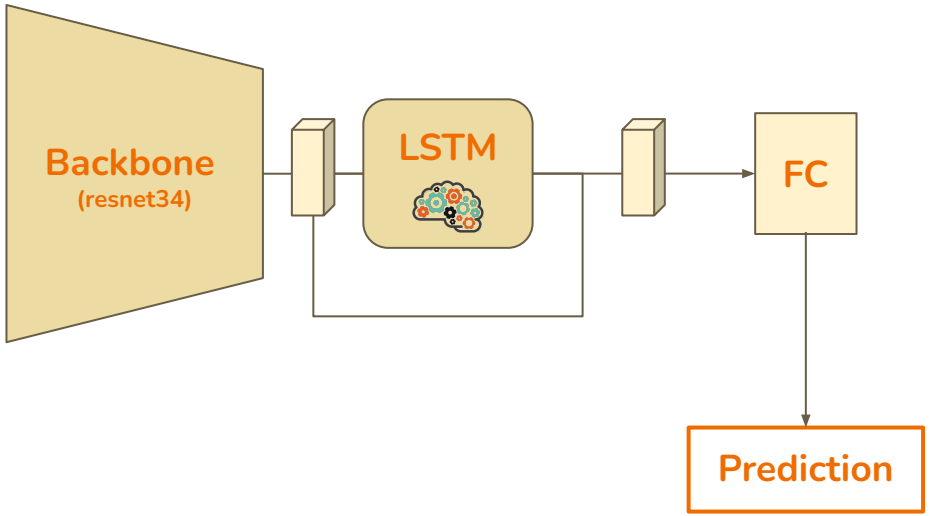
*Learning without Temporal Information (avg)*

# Network - 0

# Network - 1

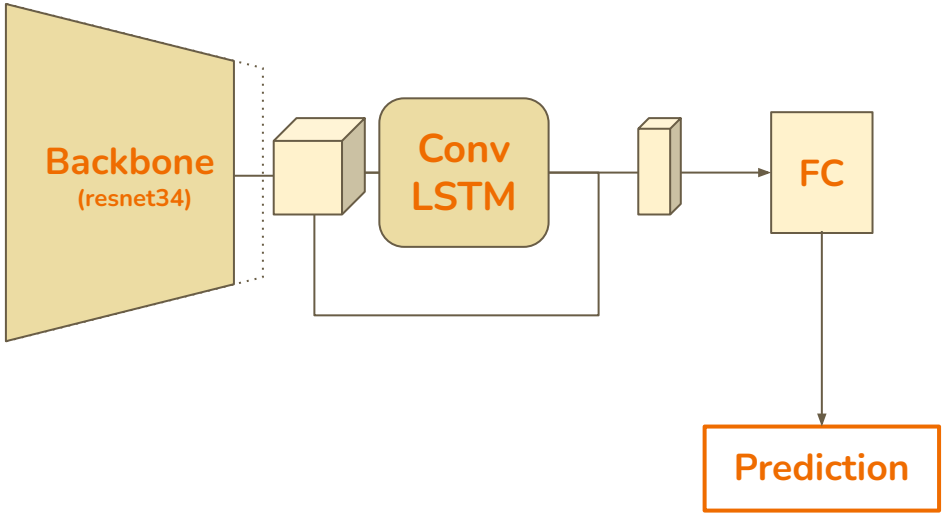
# Network - 1

# Network - 2

*Learning with Spatio-Temporal Information (ConvLSTM)*

# Network - 2

# Dataset: GTEA61

# Dataset:GTEA61

- 61 actions
- 4 users:
  - S1, S2, S3, S4

Training sets = S1 S3 S4 (labeled)

Validation set = Test set = S2.

Dataset:
https://drive.google.com/drive/folders/1_NAcoR0UGH1eLsiWMOx_Py8yeAocknA2?usp=sharing

# Code templates

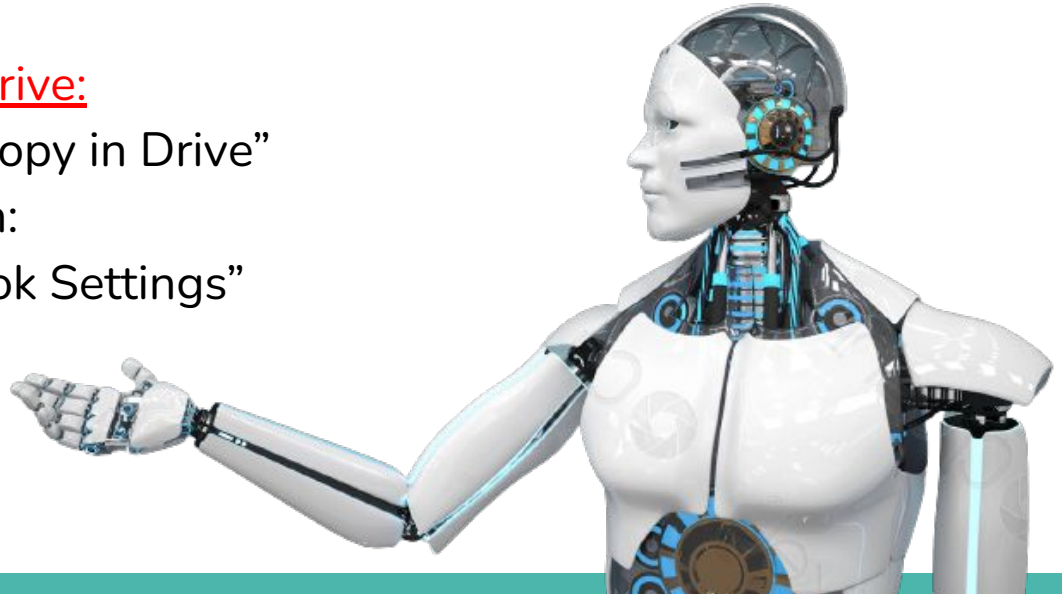
The template of the main code is available here:

Save a copy on your own drive:

"File" -> "Save a copy in Drive"

Switch to GPU acceleration:

"Edit" -> "Notebook Settings"



# Step 0: Before you start

*Learning without Temporal Information (avg)*

# Before you start

1. Study code and data:
    a. Read carefully the template code (including the comments) to understand how everything is done.
    b. Explore the data provided.

2. Run the code:
    a. try to run the code, “Learning without Temporal Information” is already implemented
    b. you have to stay connected

# Step 1: LSTM

*Learning with Temporal Information (LSTM)*

# Step 1: LSTM

1. Set the variable *homework_step = 1* in MAIN PARAMs
2. Implement the **LSTM** in the class *MyLSTM*, section *Model*
   a. *you should implement the model in the init function and the forward*
3. *Train only the **LSTM** and the **Classifier** (maintaining freezed the backbone)*
   a. *follow the same procedure used in Build Model - Loss - Opt and in Training to train only the classifier in the Step 0*

**Hint**: Use the LSTM with frame_feat

# Step 2: ConvLSTM

*Learning with Spatio-Temporal Information (ConvLSTM)*

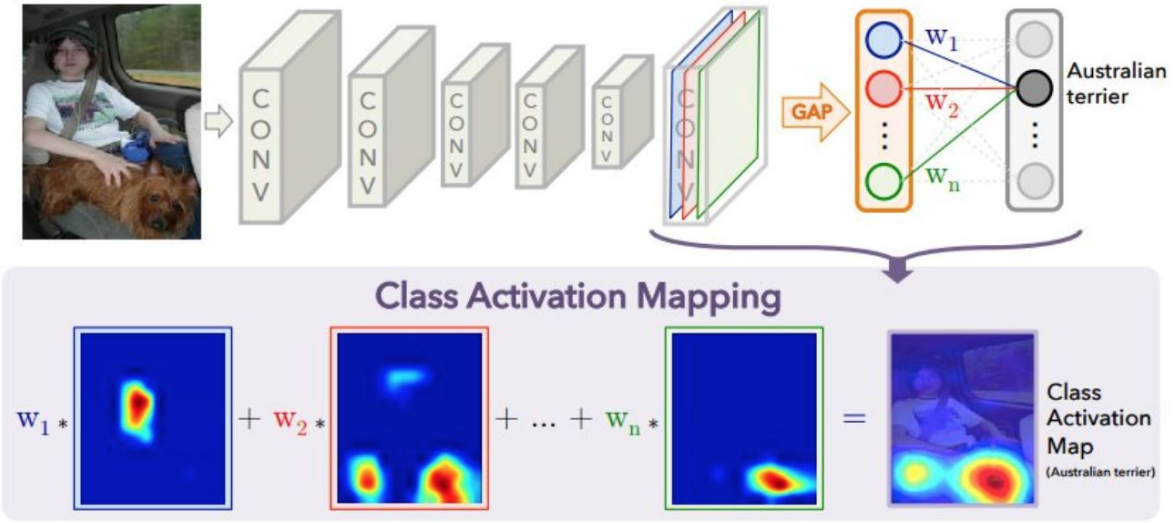
# Step 2: ConvLSTM

1. Set the variable *homework_step = 2* in MAIN PARAMs
2. Implement the **ConvLSTM** in the class *MyConvLSTM*, section *Model*
   a. *it is very similar to the standard LSTM with the difference that it use the convolution operation instead of nn.linear and it works on spatial_frames_feat instead of frames_feat*
3. *Feed into the **ConvLSTM** the features before the avgpool of the resnet34*
   a. *named spatial_frame_feat*
4. *Load the pretrained-weights that you find in "/content/best_model_state_dict_rgb_split2.pth"*
5. *Train <u>only</u> the **ConvLSTM** and the **Classifier**, maintaining freezed the backbone*

# Step 4: Class Activation Map (CAM)

Extra!

# Step 4: Class Activation Map (CAM)




Bolei Zhou, et al. “Learning Deep Features for Discriminative Localization”

# Step 4: Class Activation Map (CAM)

```
logit, feature_conv,  _ = self.resNet(inputVariable[t])

bz, nc, h, w = feature_conv.size()

feature_conv1 = feature_conv.view(bz, nc, h*w)

probs, idxs = logit.sort(1, True)

class_idx = idxs[:, 0]

cam =
torch.bmm(self.weight_softmax[class_idx].unsqueeze(1),
feature_conv1)
```

```
cam_img = F.softmax(cam, 1).data

cam_img = cam_img.cpu().numpy()

cam_img = cam_img.reshape(h, w)

cam_img = cam_img - np.min(cam_img)

cam_img = cam_img / np.max(cam_img)

cam_img = np.uint8(255 * cam_img)

output_cam = cv2.resize(cam_img, size_upsample)

img = cv2.cvtColor(np.uint8(img), cv2.COLOR_RGB2BGR)

heatmap = cv2.applyColorMap(output_cam, cv2.COLORMAP_JET)

result = heatmap * 0.3 + img * 0.5
```

# Now it's your turn, try!

Special thanks to Mirco Planamente for sharing the material