IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Motion Magnification

Reminder

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Reminder

Example: effect of the wind on a crane

Reminder

Problem statement

Original images :

$$I(x, t0) = f(x)$$
$$I(x, t1) = f(x+\delta(t1))$$

→

Magnified image :

$$I'(x, t1) = f(x + (1 + \alpha)\delta(t))$$

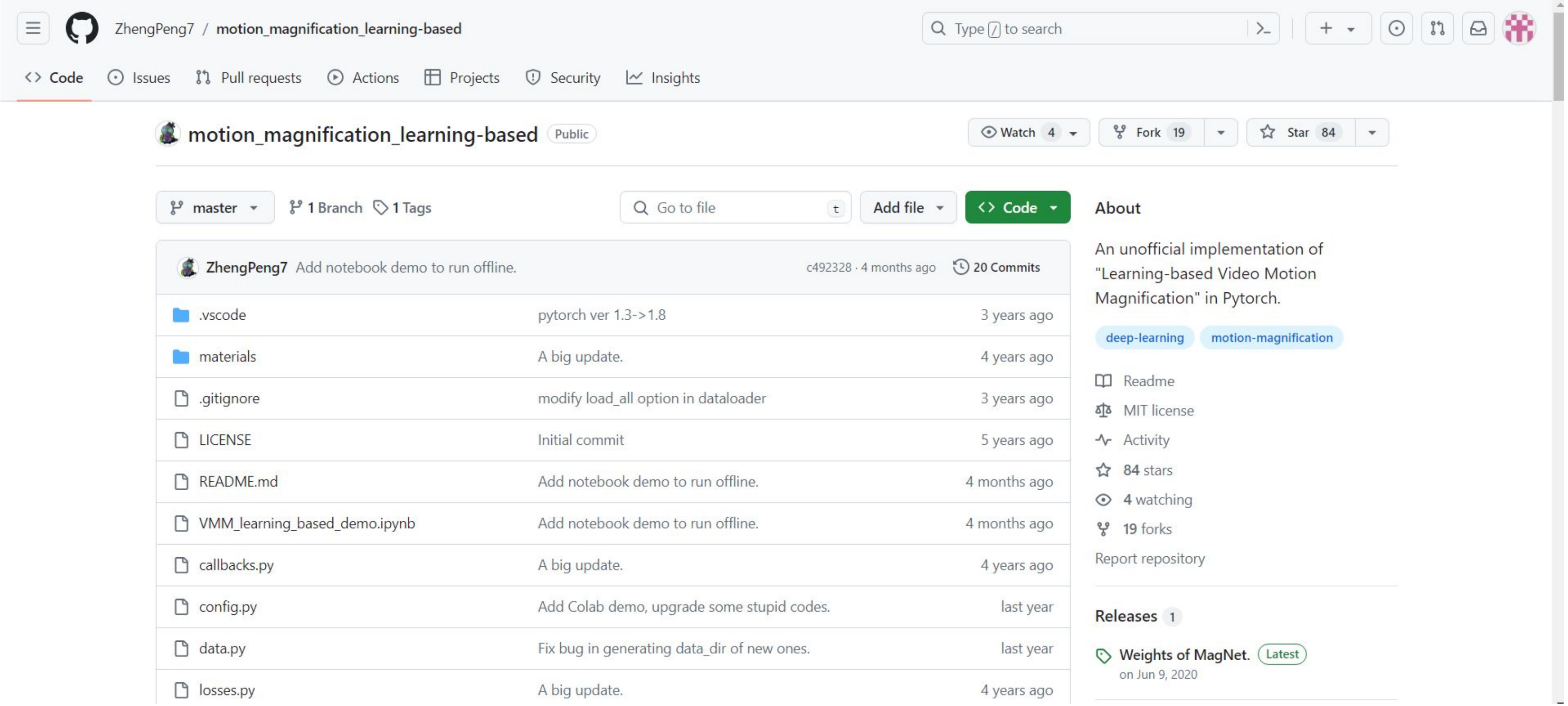
- $\delta(t)$ = the motion field
- α = magnification factor

Implementation

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Source code

https://github.com/ZhengPeng7/motion_magnification_learning-based



motion_magnification_learning-based Public

ZhengPeng7 Add notebook demo to run offline. c492328 · 4 months ago 20 Commits

.vscode	pytorch ver 1.3->1.8	3 years ago
materials	A big update.	4 years ago
.gitignore	modify load_all option in dataloader	3 years ago
LICENSE	Initial commit	5 years ago
README.md	Add notebook demo to run offline.	4 months ago
VMM_learning_based_demo.ipynb	Add notebook demo to run offline.	4 months ago
callbacks.py	A big update.	4 years ago
config.py	Add Colab demo, upgrade some stupid codes.	last year
data.py	Fix bug in generating data_dir of new ones.	last year
losses.py	A big update.	4 years ago

About

An unofficial implementation of "Learning-based Video Motion Magnification" in Pytorch.

deep-learning motion-magnification

- Readme
- MIT license
- Activity
- 84 stars
- 4 watching
- 19 forks

Report repository

Releases 1

Weights of MagNet. Latest
on Jun 9, 2020

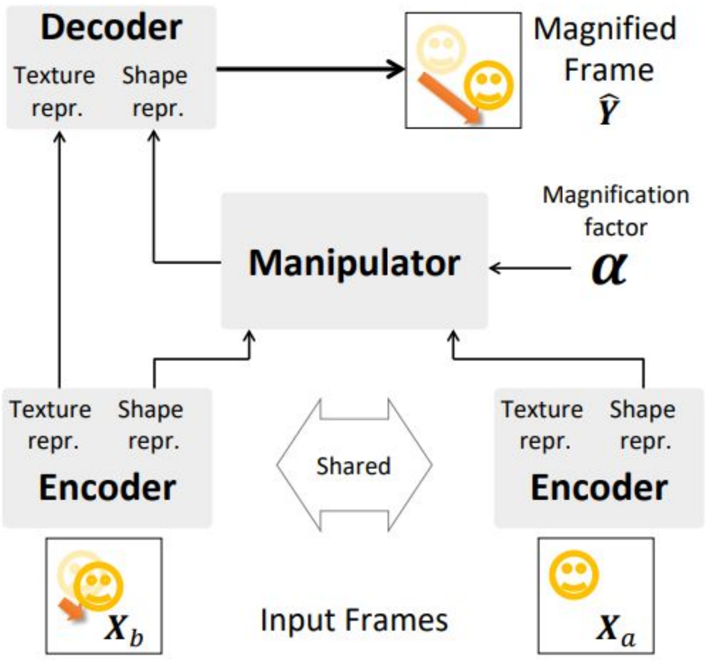
Our algorithm: Mag Net

Applying 2-frames setting to videos:

Input: 2 Consecutive Frames of a video Xa & Xb (Dynamic mode)

Output: Magnified Frame Ŷ

Overview of architecture



Decoder

Texture repr.

Shape repr.

Magnified Frame $\hat{Y}$

Manipulator

Magnification factor α

Texture repr.

Shape repr.

Encoder

Shared

Texture repr.

Shape repr.

Encoder

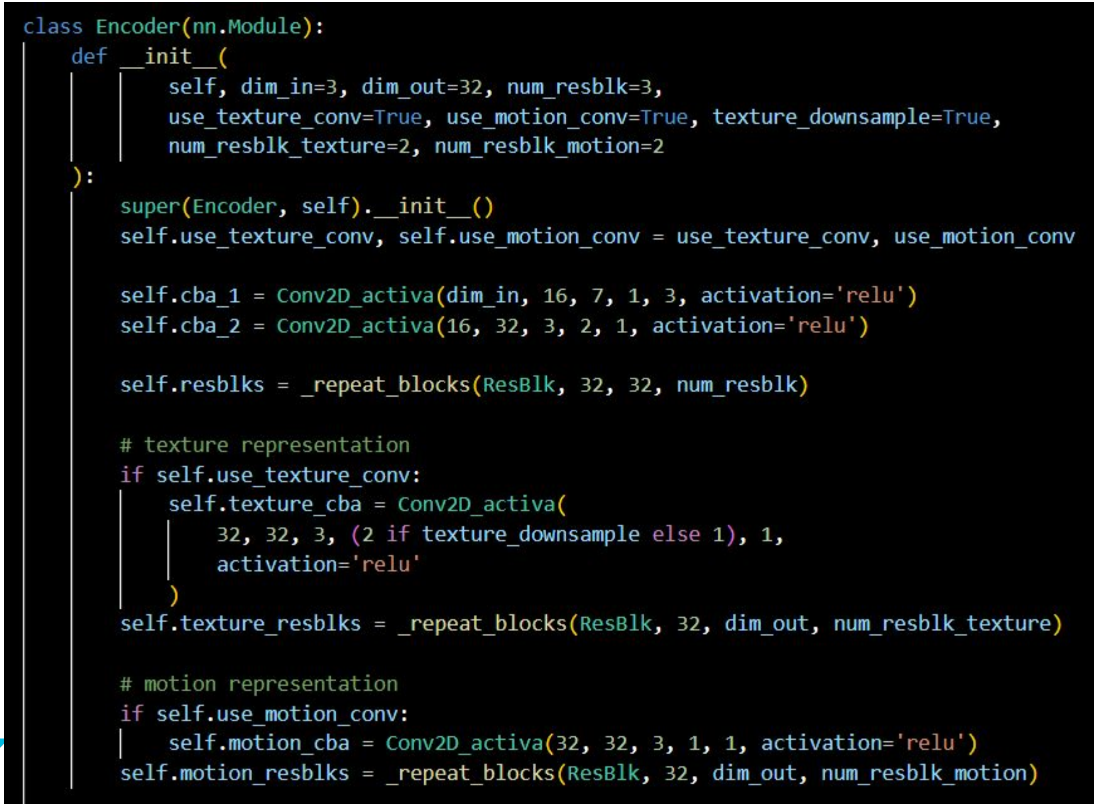
X_b

Input Frames

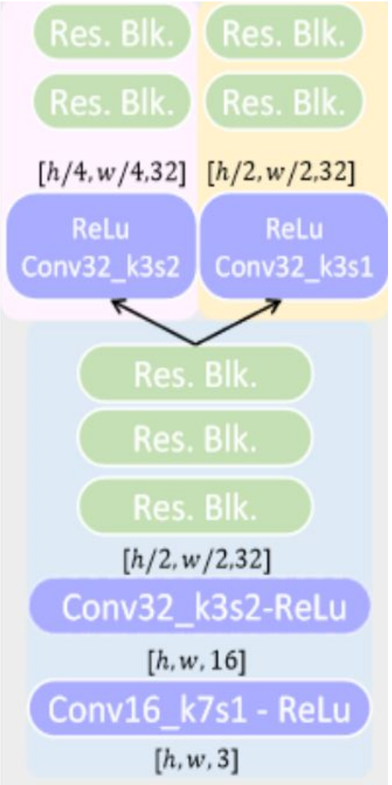
X_a

(a)

Encoder

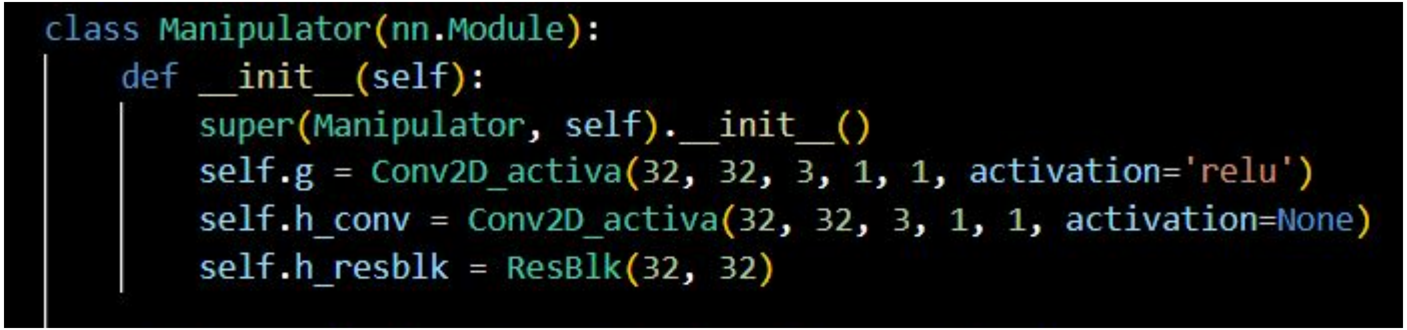


```python
class Encoder(nn.Module):
    def __init__(
            self, dim_in=3, dim_out=32, num_resblk=3,
            use_texture_conv=True, use_motion_conv=True, texture_downsample=True,
            num_resblk_texture=2, num_resblk_motion=2
    ):
        super(Encoder, self).__init__()
        self.use_texture_conv, self.use_motion_conv = use_texture_conv, use_motion_conv

        self.cba_1 = Conv2D_activa(dim_in, 16, 7, 1, 3, activation='relu')
        self.cba_2 = Conv2D_activa(16, 32, 3, 2, 1, activation='relu')

        self.resblks = _repeat_blocks(ResBlk, 32, 32, num_resblk)

        # texture representation
        if self.use_texture_conv:
            self.texture_cba = Conv2D_activa(
                32, 32, 3, (2 if texture_downsample else 1), 1,
                activation='relu'
            )
        self.texture_resblks = _repeat_blocks(ResBlk, 32, dim_out, num_resblk_texture)

        # motion representation
        if self.use_motion_conv:
            self.motion_cba = Conv2D_activa(32, 32, 3, 1, 1, activation='relu')
        self.motion_resblks = _repeat_blocks(ResBlk, 32, dim_out, num_resblk_motion)
```

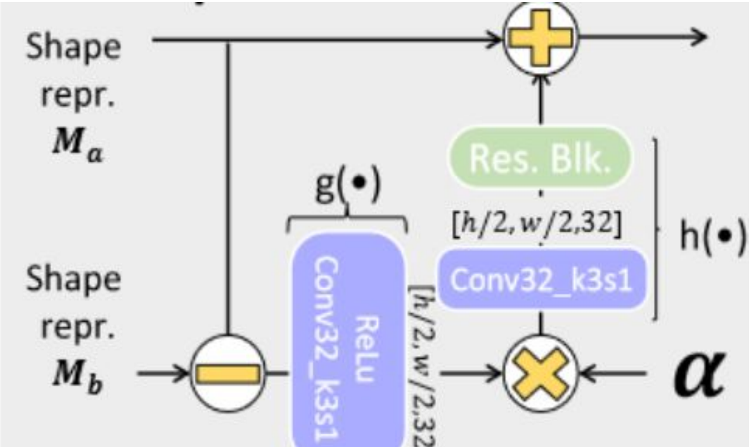


Manipulator

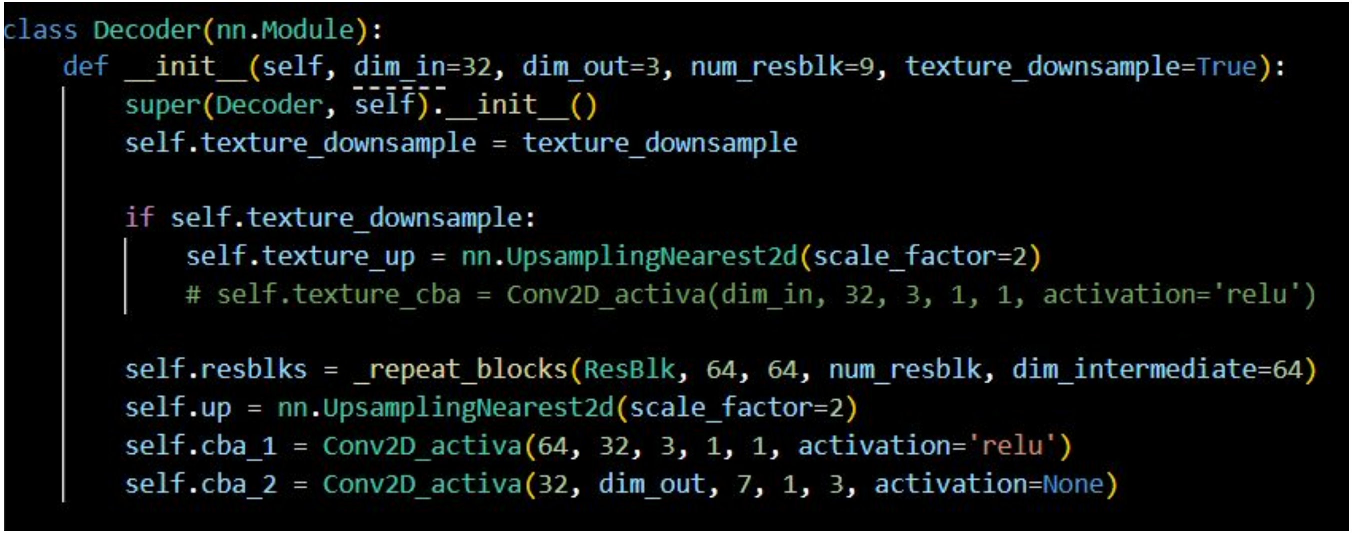
$$G_m(\mathbf{M}_a, \mathbf{M}_b, \alpha) = \mathbf{M}_a + h\left(\alpha \cdot g(\mathbf{M}_b - \mathbf{M}_a)\right)$$



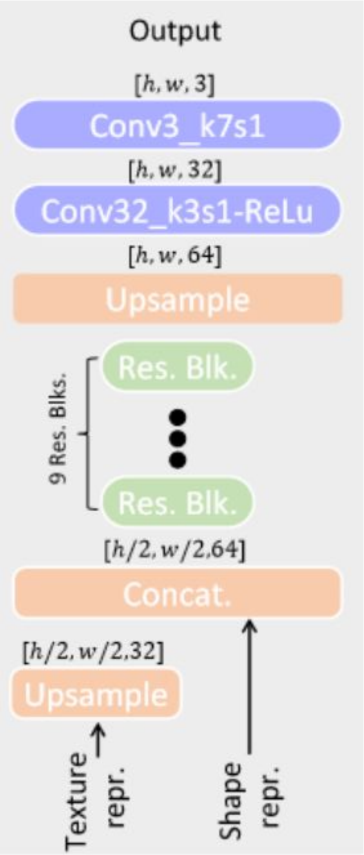
```
class Manipulator(nn.Module):
    def __init__(self):
        super(Manipulator, self).__init__()
        self.g = Conv2D_activa(32, 32, 3, 1, 1, activation='relu')
        self.h_conv = Conv2D_activa(32, 32, 3, 1, 1, activation=None)
        self.h_resblk = ResBlk(32, 32)
```



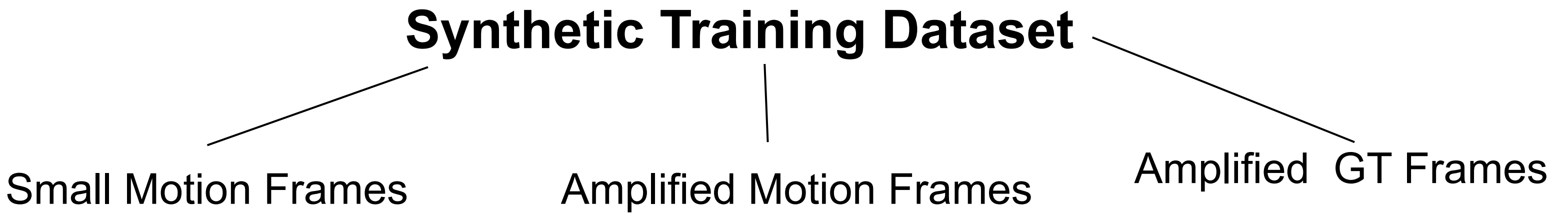
Decoder



```
class Decoder(nn.Module):
    def __init__(self, dim_in=32, dim_out=3, num_resblk=9, texture_downsample=True):
        super(Decoder, self).__init__()
        self.texture_downsample = texture_downsample

        if self.texture_downsample:
            self.texture_up = nn.UpsamplingNearest2d(scale_factor=2)
            # self.texture_cba = Conv2D_activa(dim_in, 32, 3, 1, 1, activation='relu')

        self.resblks = _repeat_blocks(ResBlk, 64, 64, num_resblk, dim_intermediate=64)
        self.up = nn.UpsamplingNearest2d(scale_factor=2)
        self.cba_1 = Conv2D_activa(64, 32, 3, 1, 1, activation='relu')
        self.cba_2 = Conv2D_activa(32, dim_out, 7, 1, 3, activation=None)
```



Data set



Synthetic Training Dataset

Small Motion Frames

Amplified Motion Frames

Amplified GT Frames

- *Background:* 200,000 images from *MS COCO dataset.*
- *Foreground:* 7,000 segmented objects from the PASCAL VOC dataset.

Training

Objective: Minimize the loss function during training, using L1-loss.
Optimizer: ADAM optimizer is employed for its effectiveness in optimizing deep learning models.

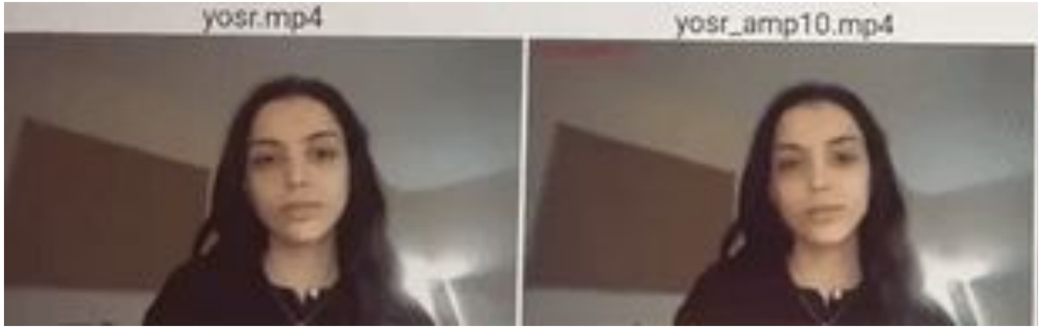
Parameters	
β1	0.9
β2	0.999
Learning Rate	1e-4
Batch Size	4

Results

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

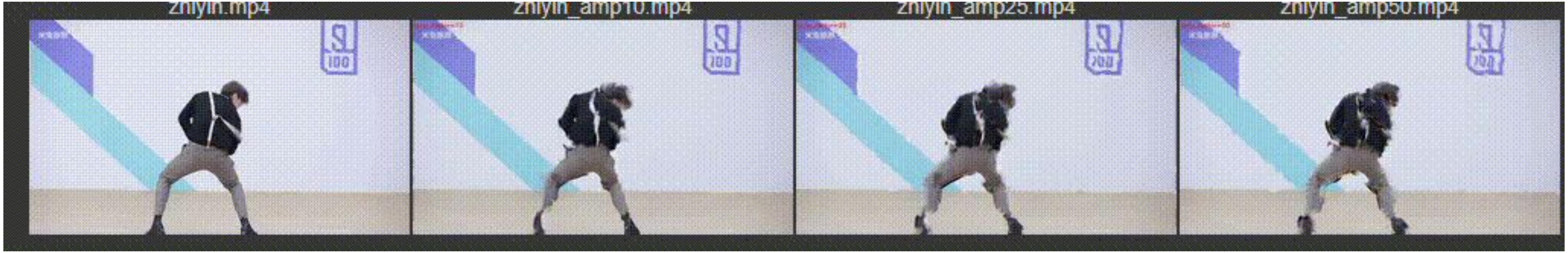
Results

Pretrained Model for our tests: Number of epochs: 12 - Loss: 7.28e-02



Magnification factor influence

Performance degradation with high α

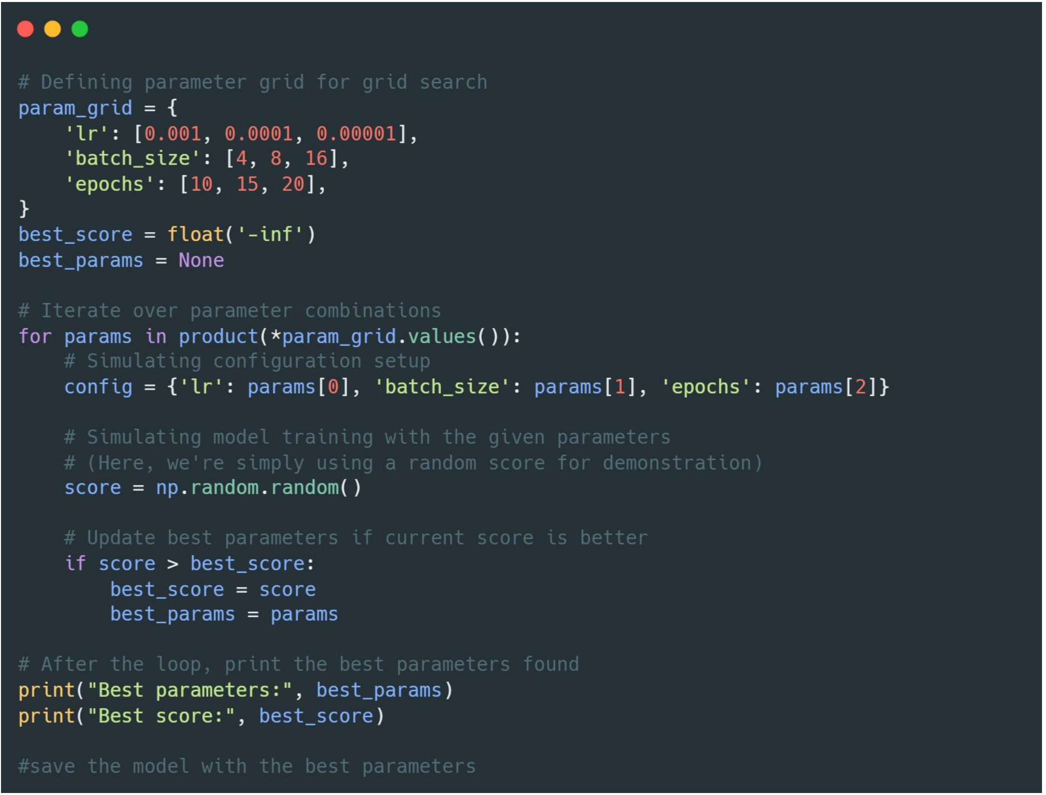


Issue: Blurring and color artifacts with high magnification factors.
→ Limit magnification factor (α) up to 100.

Encountered issues

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

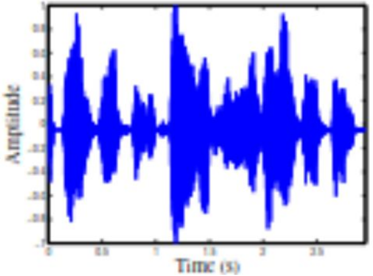
Grid search



```python
# Defining parameter grid for grid search
param_grid = {
    'lr': [0.001, 0.0001, 0.00001],
    'batch_size': [4, 8, 16],
    'epochs': [10, 15, 20],
}
best_score = float('-inf')
best_params = None

# Iterate over parameter combinations
for params in product(*param_grid.values()):
    # Simulating configuration setup
    config = {'lr': params[0], 'batch_size': params[1], 'epochs': params[2]}

    # Simulating model training with the given parameters
    # (Here, we're simply using a random score for demonstration)
    score = np.random.random()

    # Update best parameters if current score is better
    if score > best_score:
        best_score = score
        best_params = params

# After the loop, print the best parameters found
print("Best parameters:", best_params)
print("Best score:", best_score)

#save the model with the best parameters
```

High computational cost !

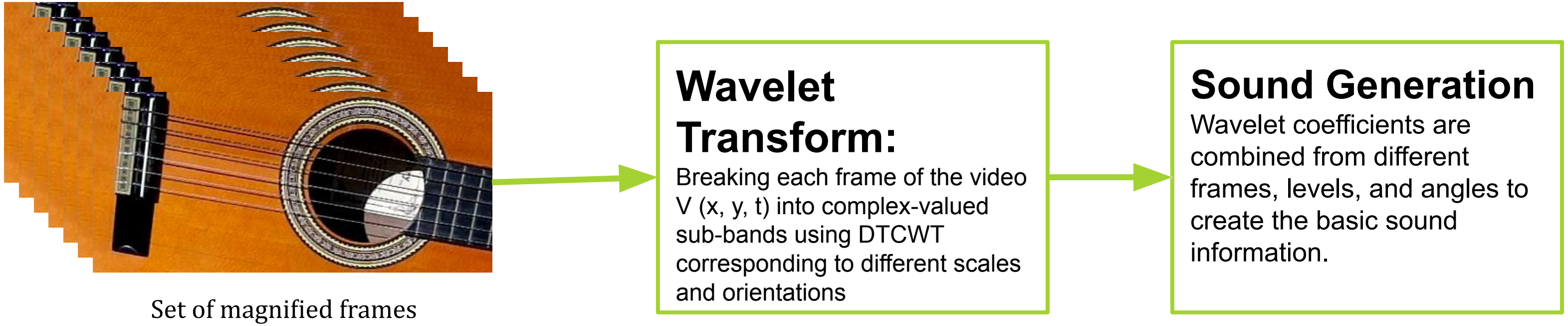
Sound Extraction



Desired output

- Objects react differently to sound vibrations.
- Factors influencing these vibrations:
  - Material of the object.
  - Frequency of the sound.
  - Distance from the sound source.
  - Edge direction of the object.

The pipeline



Set of magnified frames

Wavelet Transform:
Breaking each frame of the video V (x, y, t) into complex-valued sub-bands using DTCWT corresponding to different scales and orientations

Sound Generation
Wavelet coefficients are combined from different frames, levels, and angles to create the basic sound information.

Thank you for you attention

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom