### Видео

video\_stream = cv2.VideoCapture(args.face)  
fps = video\_stream.get(cv2.CAP\_PROP\_FPS)  
  
print('Reading video frames...')  
  
full\_frames = []  
while 1:  
 still\_reading, frame = video\_stream.read()  
 if not still\_reading:  
 video\_stream.release()  
 break  
 if args.resize\_factor > 1:  
 frame = cv2.resize(frame, (frame.shape[1]//args.resize\_factor, frame.shape[0]//args.resize\_factor))  
  
 if args.rotate:  
 frame = cv2.rotate(frame, cv2.cv2.ROTATE\_90\_CLOCKWISE)  
  
 y1, y2, x1, x2 = args.crop  
 if x2 == -1: x2 = frame.shape[1]  
 if y2 == -1: y2 = frame.shape[0]  
  
 frame = frame[y1:y2, x1:x2]  
  
 full\_frames.append(frame)

### Аудио

def load\_wav(path, sr):  
 return librosa.core.load(path, sr=sr)[0]

wav = audio.load\_wav(args.audio, 16000)  
mel = audio.melspectrogram(wav)

Audio filtering

def preemphasis(wav, k, preemphasize=True):  
 if preemphasize:  
 return signal.lfilter([1, -k], [1], wav)  
 return wav

def \_stft(y):  
 if hp.use\_lws:  
 return \_lws\_processor(hp).stft(y).T  
 else:  
 return librosa.stft(y=y, n\_fft=hp.n\_fft, hop\_length=get\_hop\_size(), win\_length=hp.win\_size)

SPECTROGRAM

**librosa.filters.mel**

Mel filter bank construction

**librosa.stft**

Short-time Fourier Transform

def melspectrogram(wav):  
 D = \_stft(preemphasis(wav, hp.preemphasis, hp.preemphasize))  
 S = \_amp\_to\_db(\_linear\_to\_mel(np.abs(D))) - hp.ref\_level\_db  
   
 if hp.signal\_normalization:  
 return \_normalize(S)  
 return S

### 