Dense Video Captioning Lite

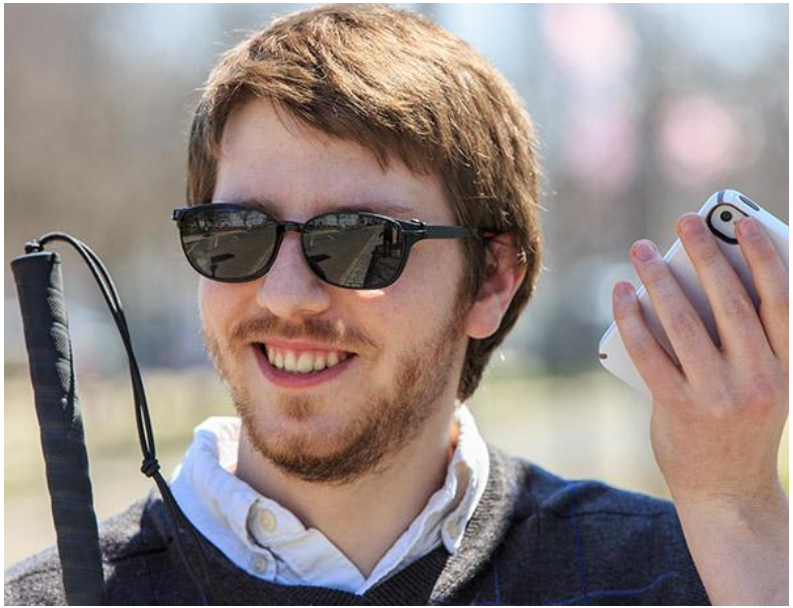
Abhishek Jain, Maher Khan & Swapnil Asawa

Problem we are trying to solve(Recap)

Problem: Real time video captioning on mobile devices

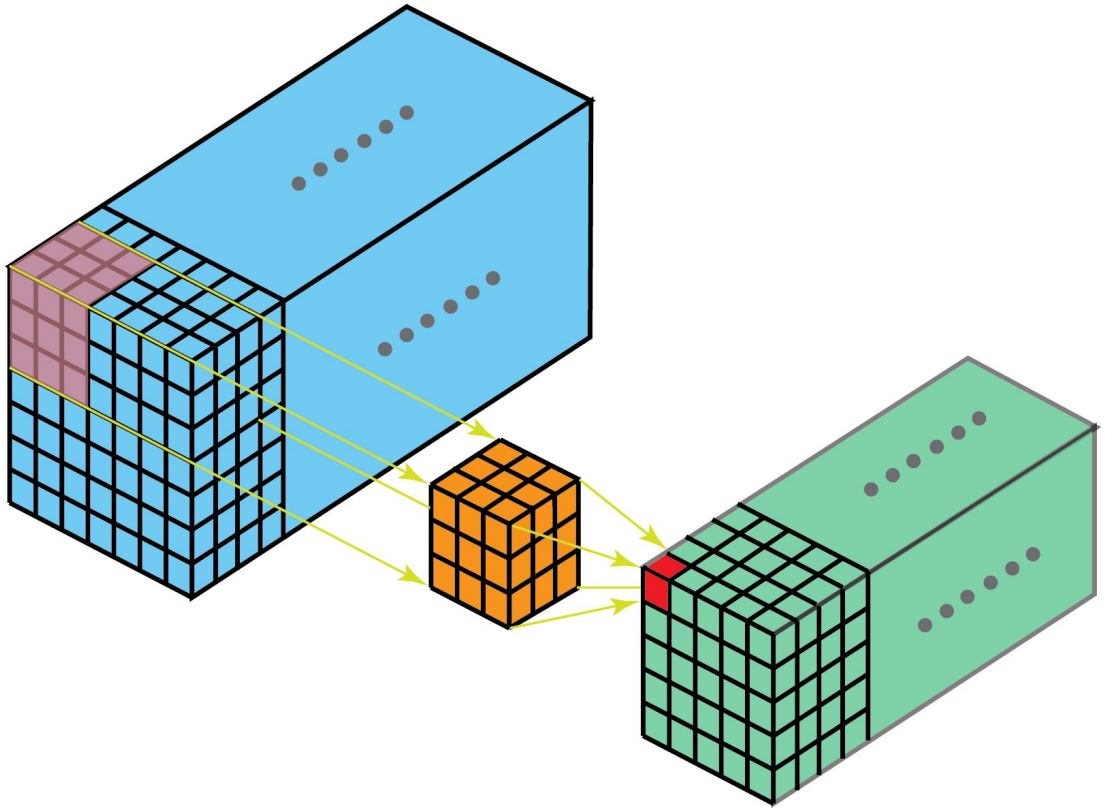
- Natural language descriptions of videos
- Fast and efficient processing of videos
- Portable

Example: A blind person trying to understand what is **happening** around him using a smartphone.



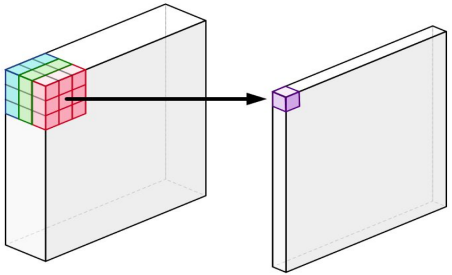
What are we doing differently?

3D CNN:

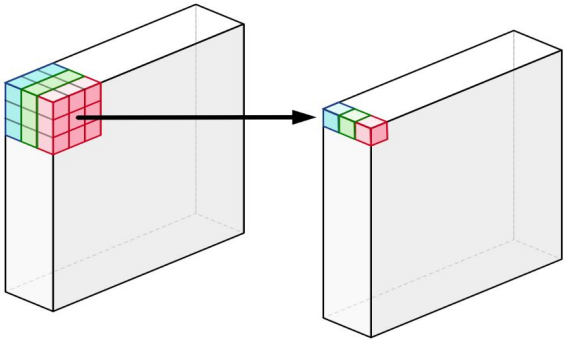


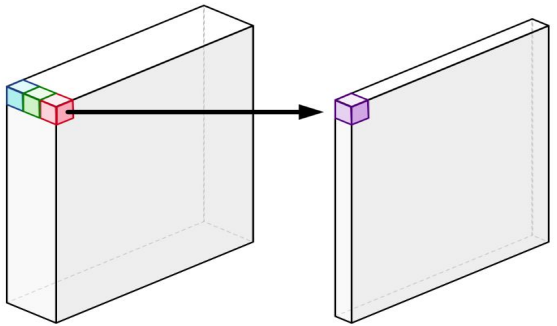
There is a better way to convolve: MobileNetV2

Normal Convolution:



Mobilenet Convolution:(9 times reduced computation)





MobilenetV2 is an advancement of Mobilenet

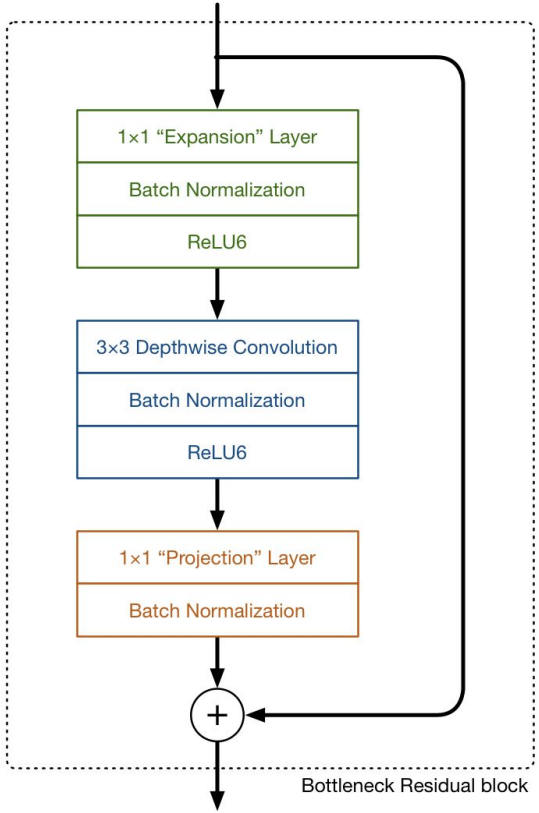
What is MobileNet V2?

Efficient architecture for 2D convolution

- Based on an inverted residual structure where the between the thin bottleneck layers.
- Only ~3.5M parameters vs 33M for C3D
- Good for Mobile devices
- Marginal loss of accuracy

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2-bottleneck	21.9	6.03	68.9M	93
pre-ResNet-200	21.66	5.79	64.7M	154

Table 8: ILSVRC-2012 validation error (single crop) of bottleneck ResNets. Faster WRN-50-2-bottleneck outperforms ResNet-152 having 3 times less layers, and stands close to pre-ResNet-200.



Implementation (Part 1)

```
class BottleneckResBlock(nn.Module):
    def __init__(self, in_channel_size, out_channel_size, expansion=False, stride=1):
        super(BottleneckResBlock, self).__init__()
        self.stride = stride
        channel_size = self.in_channel_size = in_channel_size
        self.out_channel_size = out_channel_size

        if expansion:
            self.block = nn.Sequential(
                #expansion layer
                nn.Conv2d(in_channels=channel_size, out_channels=expansion*channel_size,
                                        kernel_size=1, stride=1, padding=0, bias=False),
                nn.BatchNorm2d(num_features=expansion*channel_size),
                nn.ReLU6(inplace=True),

                #"Depthwise" Convolution
                nn.Conv2d(in_channels=expansion*channel_size, out_channels=expansion*channel_size,
                                        kernel_size=3, stride=stride, padding=1, groups=expansion*channel_size, bias=False),
                nn.BatchNorm2d(num_features=expansion*channel_size),
                nn.ReLU6(inplace=True),

                #"Projection" Layer
                nn.Conv2d(in_channels=expansion*channel_size, out_channels=out_channel_size,
                                        kernel_size=1, stride=1, padding=0, bias=False),
                nn.BatchNorm2d(num_features=out_channel_size),
                )
```

```
BottleneckResBlock_config = [
    # t, c, n, s as mentioned
    [1, 16, 1, 1],
    [6, 24, 2, 2],
    [6, 32, 3, 1],
    [6, 64, 4, 1],
    [6, 96, 3, 1],
    [6, 160, 3, 2],
    [6, 320, 1, 1],
]
```

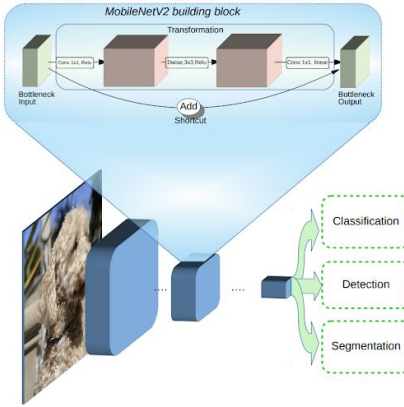
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.

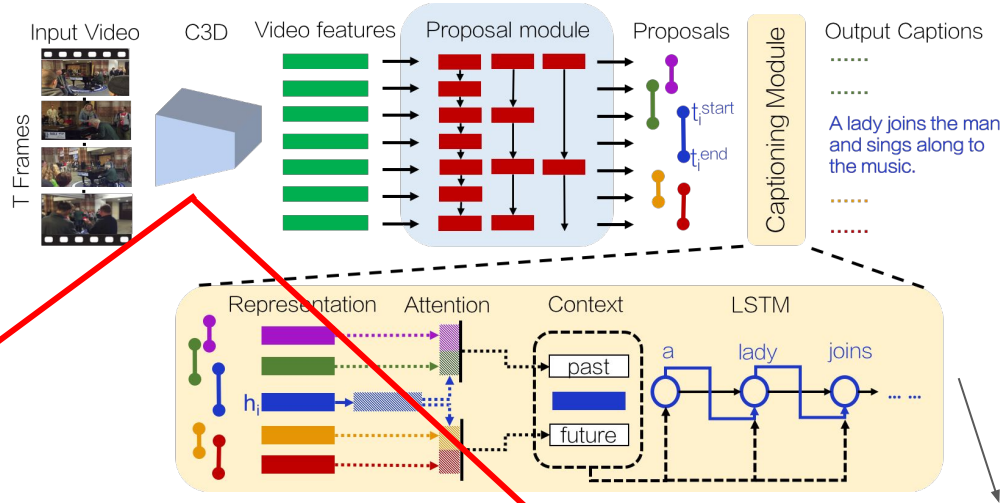
Some Hyperparameters that we tuned:

```
numEpochs = 20

learningRate = 1e-3

weightDecay = 1e-4

num_classes = len(train_dataset.classes)

network.apply(init_weights)

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(network.parameters(), lr=learningRate, weight_decay=weightDecay)
```

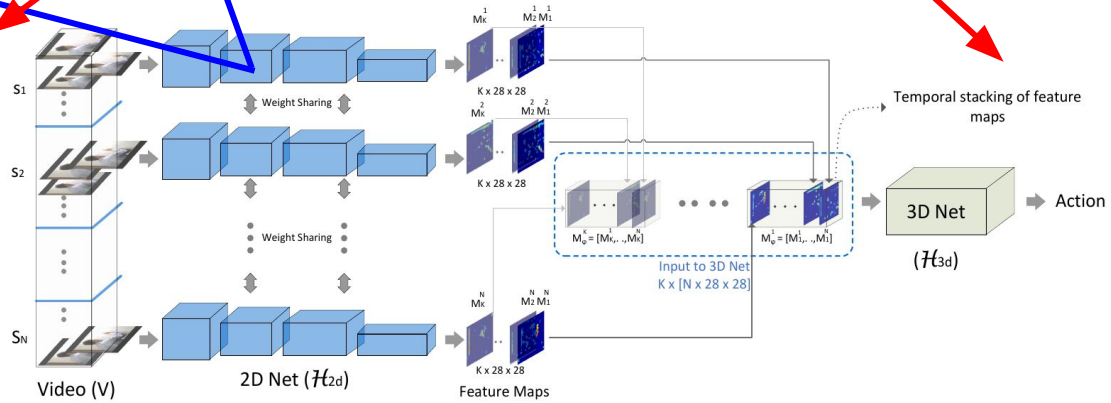
Our Overall Approach Revisited



MobileNet V2
(Sandler et al, 2019)



Dense Video
Captioning
(Krishna et al, 2



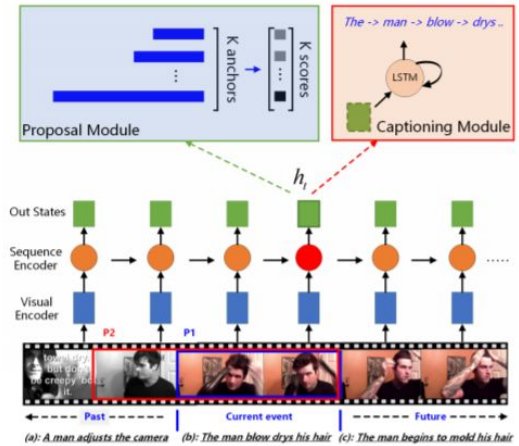
ECO Lite
(Zolfaghari et al, 2018)

Improvements over the previous approach

- Using MobileNetV2 directly instead of ECOLite in between.
  - Benefits: Faster.
  - Temporal connections are also extracted from LSTM encoder
- Bidirectional Attentive Fusion with Context Gating for Dense Video Captioning

Bidirectional Attentive Fusion with Context Gating for Dense Video Captioning

- Earlier different events ending at the same time gave same caption.
- Solved by representing each event with an attentive fusion.
- Outperforms the state-of-the-arts on the ActivityNet Captions dataset with a gain of over 100%.



Steps

- LSTM hidden state at time step t thus encodes visual information for the passed time steps {1, 2.. t}.
- Goes to K classifiers to produce K confidence scores of k predefined lengths ending at t.
- Backward proposal: must give good confidence if proposal starts at time t.
- Fuse them.
- Choose the ones with high scores.

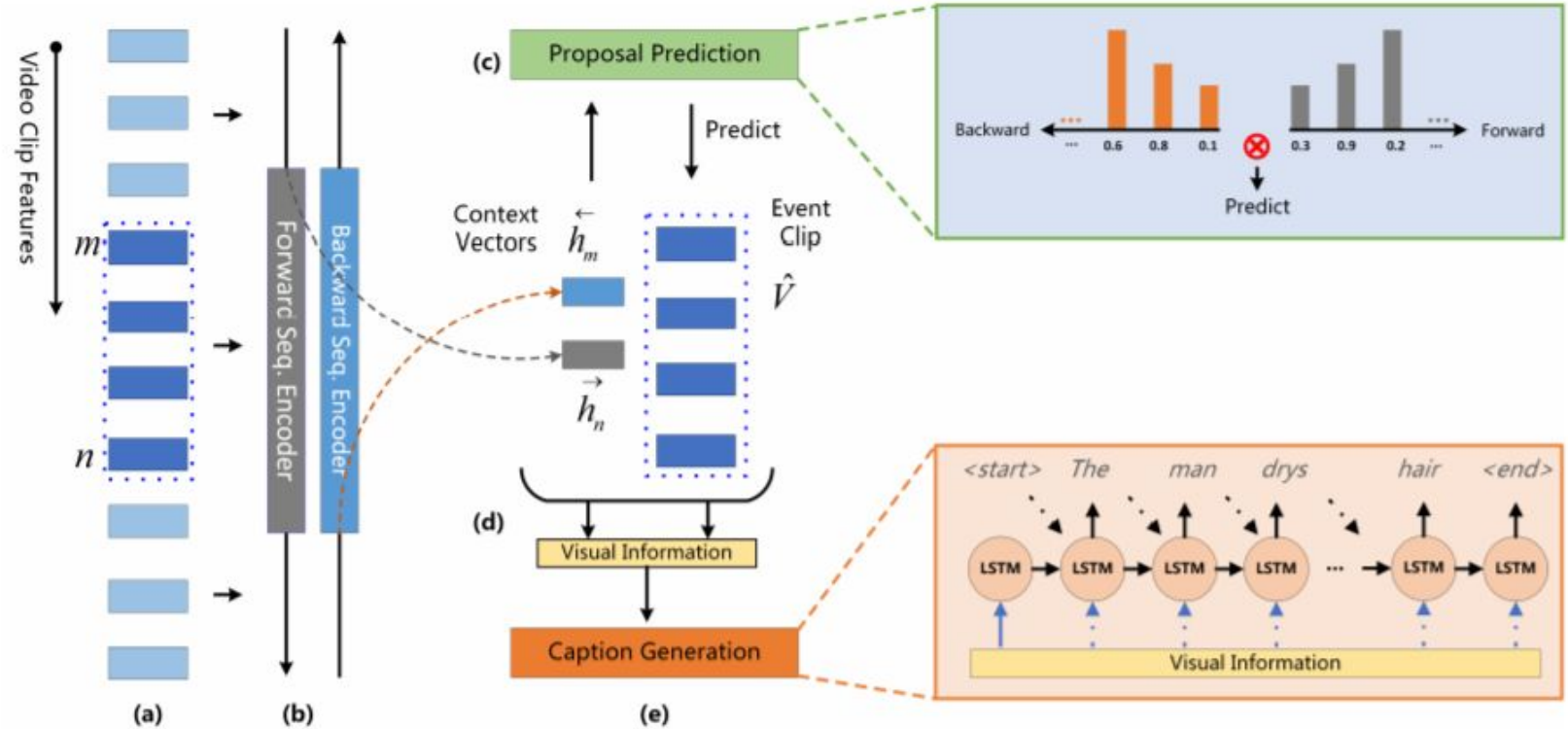


Figure 2. The main framework of our proposed method. (a) A video input is first encoded as a sequence of visual features (e.g., C3D). (b) The visual features are then fed into our bidirectional sequence encoder (e.g., LSTM). (c) Each hidden state from the forward/backward seq. encoder will be fed into the proposal module. The forward/backward seq. encoders are jointly learned to make proposal predictions. (d) Hidden states at the boundary of a detected event ($\overrightarrow{\mathbf{h}}_n$, $\overleftarrow{\mathbf{h}}_m$) will be served as context vectors for the event. The context vectors and detected event clip features are then fused together and served as visual information input. We detail the fusion methods in Section 3.2.2. (e) The decoder LSTM translates visual input into a sentence.

Our implementation continued...

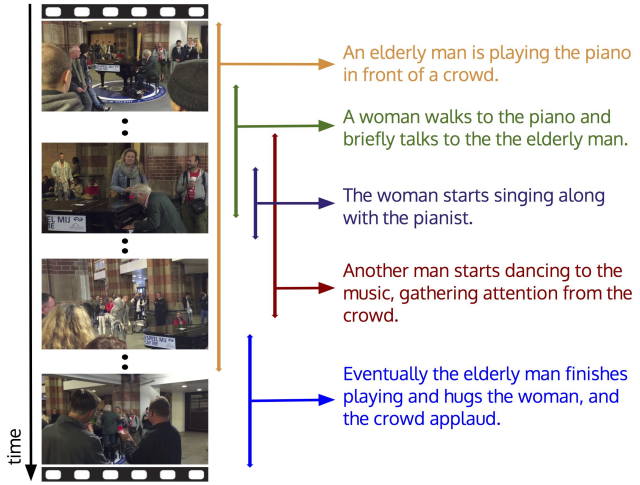
Datasets

We worked with two datasets: UCF101 and ActivityNet

UCF101

(Maps each video to a particular action)

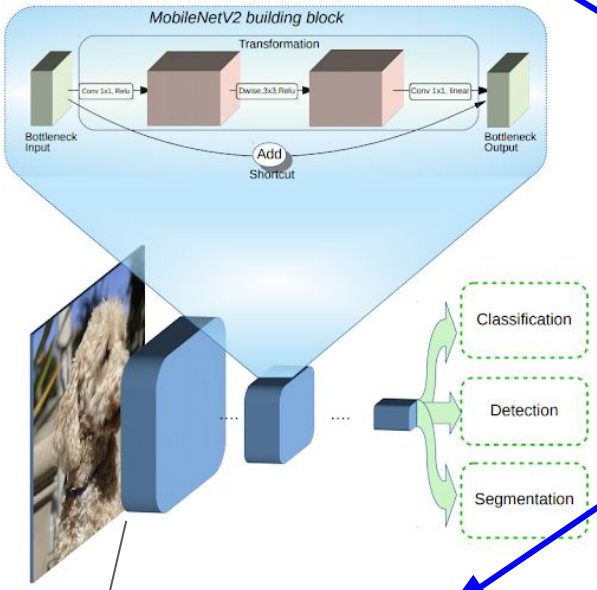
ActivityNet



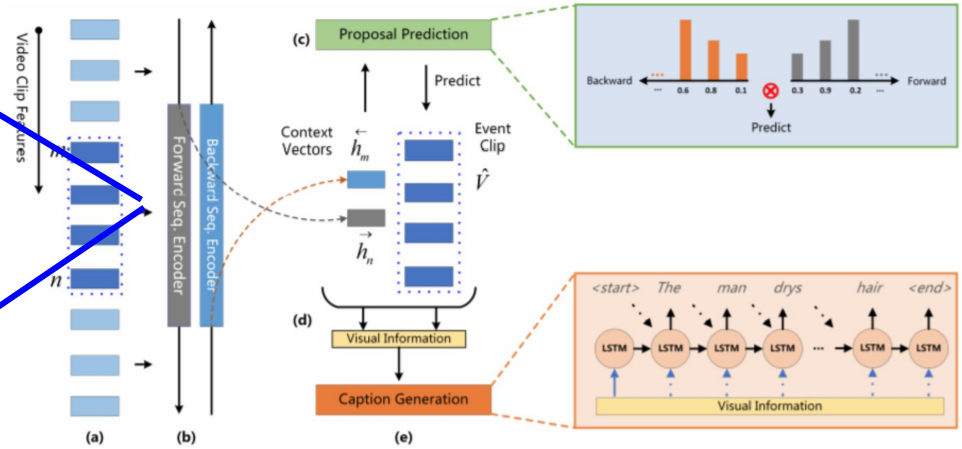
(Tags each video with overlapping captions)

Implementation (Part 2)

- Implemented our own version of MobileNetV2
- Trained MobileNetV2 model on UCF101 dataset
- Replaced 3D-CovNet in DenseVideoCaptioning architecture with MobileNetV2



MobileNet V2
(Sandler et al, 2019)

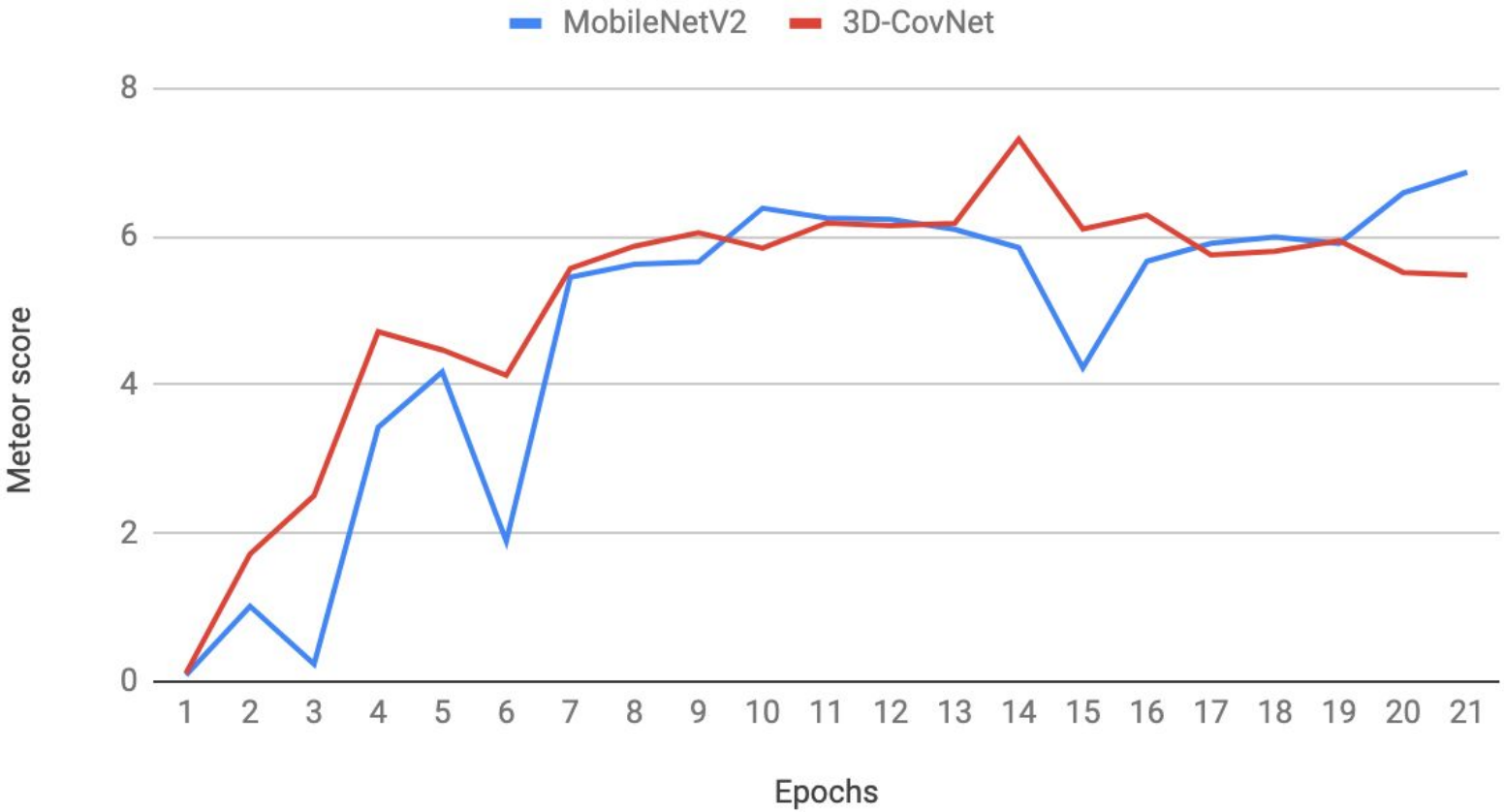


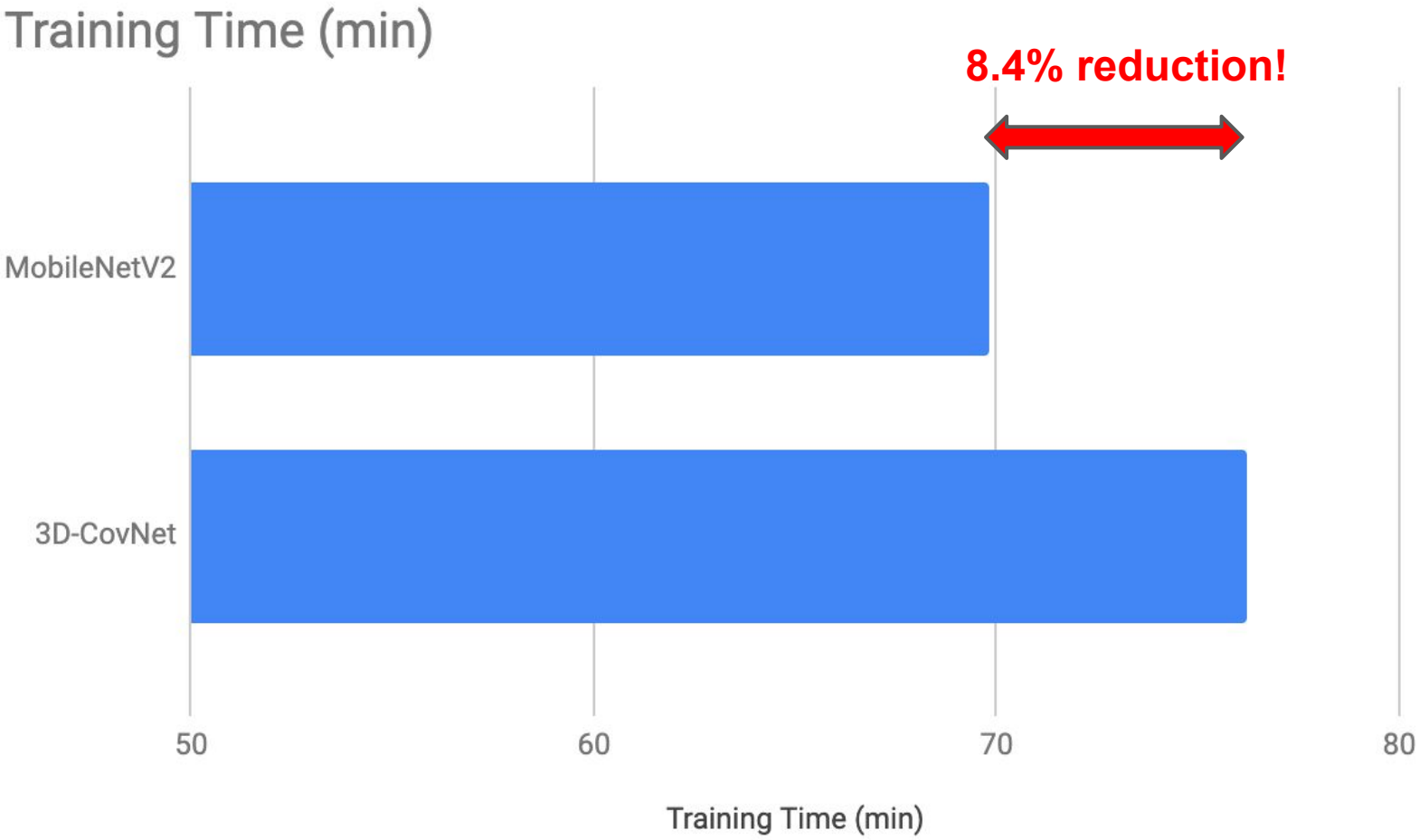
Bidirectional Dense
Video Captioning
(Wang et al, 2018)

Implementation (Part 3)

- MobileNetV2 extracts 2D frame features from ActivityNet dataset
- Trained the Proposal and Caption modules using these frame features
- Tested the new model for accuracy and speed

Training Accuracy




Training Time (min)
8.4% reduction!
MobileNetV2
3D-CovNet
50
60
70
80
Training Time (min)

Example of a caption

Ground Truth: “A man is seen kneeling down on the floor speaking to the camera”

MobileNetV2: “A man is seen shown in a camera”

3D-CovNet: “A man is seen is standing in a stage”

Analysis

- We were right!
- Temporal aspect is already captured by the Proposal module.
- So, 3D-Covnet is providing redundant information
- Thus, MobileNetV2 is enough!

Number of Parameters Improvement

- 3D-CovNet: 33.00 Million parameters
- MobNetV2: 3.47 Million parameters
- 1/10 reduction of parameters!
- Allow to implement on mobile devices for real time dense captioning

Limitations:

- ActivityNet has 20,000 videos!
- Total frames = 2,250,000! ((20000 * 5 * 60 * 24) / 64)
- Limited by hardware resources
- Limited by training time

Future Work

- Further experimentations with MobileNetV2 architecture and number of layers
- Further experimentations with DenseVideoCaptioning architecture
- Train a GAN to generate sentences and use the generator of Trained GAN in this architecture to give captions.
- Document our approach in a paper

QUESTIONS?

References

[1] Su, Jiaqi. "Study of Video Captioning Problem."

[2] Rivera-Soto, R. A., & Ordónez, J. Sequence to Sequence Models for Generating Video Captions.

[3] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks.

[4] Zolfaghari, M., Singh, K., & Brox, T. (2018). Eco: Efficient convolutional network for online video understanding.

[5] Krishna, R., Hata, K., Ren, F., Fei-Fei, L., & Carlos Niebles, J. (2017). Dense-captioning events in videos

[6]Jingwen Wang et. al., Bidirectional Attentive Fusion with Context Gating for Dense Video Captioning.