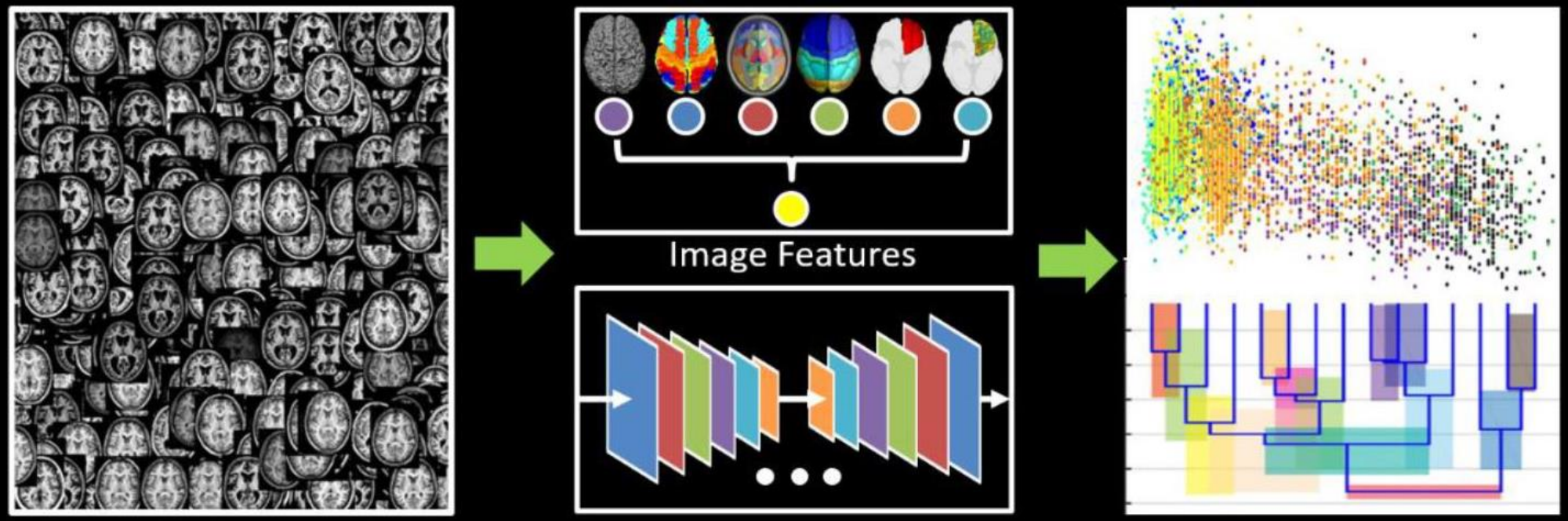


Attention

[Spring 2020 CS-8395 Deep Learning in Medical Image Computing]

Instructor: Yuankai Huo, Ph.D.
Department of Electrical Engineering and Computer Science
Vanderbilt University

About HW 3

Channels

```
self.conv1_input =      nn.Conv2d(1, 64, 3, padding=1)
self.conv1 =            nn.Conv2d(64, 64, 3, padding=1)
self.conv2_input =      nn.Conv2d(64, 128, 3, padding=1)
self.conv2 =            nn.Conv2d(128, 128, 3, padding=1)
self.conv3_input =      nn.Conv2d(128, 256, 3, padding=1)
self.conv3 =            nn.Conv2d(256, 256, 3, padding=1)
self.conv4_input =      nn.Conv2d(256, 512, 3, padding=1)
self.conv4 =            nn.Conv2d(512, 512, 3, padding=1)
self.conv5_input =      nn.Conv2d(512, 1024, 3, padding=1)
self.conv5 =            nn.Conv2d(1024, 1024, 3, padding=1)

self.conv6_up =         nn.ConvTranspose2d(1024, 512, 2, 2)
self.conv6_input =      nn.Conv2d(1024, 512, 3, padding=1)
self.conv6 =            nn.Conv2d(512, 512, 3, padding=1)
self.conv7_up =         nn.ConvTranspose2d(512, 256, 2, 2)
self.conv7_input =      nn.Conv2d(512, 256, 3, padding=1)
self.conv7 =            nn.Conv2d(256, 256, 3, padding=1)
self.conv8_up =         nn.ConvTranspose2d(256, 128, 2, 2)
self.conv8_input =      nn.Conv2d(256, 128, 3, padding=1)
self.conv8 =            nn.Conv2d(128, 128, 3, padding=1)
self.conv9_up =         nn.ConvTranspose2d(128, 64, 2, 2)
self.conv9_input =      nn.Conv2d(128, 64, 3, padding=1)
self.conv9 =            nn.Conv2d(64, 64, 3, padding=1)
self.conv9_output =     nn.Conv2d(64, 2, 1)
```

```
class UNet3D(nn.Module):
    def __init__(self, in_channel, n_classes):
        self.in_channel = in_channel
        self.n_classes = n_classes
        super(UNet3D, self).__init__()
        self.ec0 = self.encoder(self.in_channel, 32, bias=False, batchnorm=False)
        self.ec1 = self.encoder(32, 64, bias=False, batchnorm=False)
        self.ec2 = self.encoder(64, 64, bias=False, batchnorm=False)
        self.ec3 = self.encoder(64, 128, bias=False, batchnorm=False)
        self.ec4 = self.encoder(128, 128, bias=False, batchnorm=False)
        self.ec5 = self.encoder(128, 256, bias=False, batchnorm=False)
        self.ec6 = self.encoder(256, 256, bias=False, batchnorm=False)
        self.ec7 = self.encoder(256, 512, bias=False, batchnorm=False)

        self.pool0 = nn.MaxPool3d(2)
        self.pool1 = nn.MaxPool3d(2)
        self.pool2 = nn.MaxPool3d(2)

        self.dc9 = self.decoder(512, 512, kernel_size=2, stride=2, bias=False)
        self.dc8 = self.decoder(256 + 512, 256, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc7 = self.decoder(256, 256, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc6 = self.decoder(256, 256, kernel_size=2, stride=2, bias=False)
        self.dc5 = self.decoder(128 + 256, 128, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc4 = self.decoder(128, 128, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc3 = self.decoder(128, 128, kernel_size=2, stride=2, bias=False)
        self.dc2 = self.decoder(64 + 128, 64, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc1 = self.decoder(64, 64, kernel_size=3, stride=1, padding=1, bias=False)
        self.dc0 = self.decoder(64, n_classes, kernel_size=1, stride=1, bias=False)
```

Load Image

```
def __getitem__(self, index):
        num_labels = self.num_labels
        sub_name = self.img_subs[index]
        x = np.zeros((1, output_z, output_x, output_y))
        img_file = self.img_files[index]
        img_3d = nib.load(img_file)
        img = img_3d.get_data()
        img = (img - img.min())/(img.max()-img.min())
        img = img*255.0
        img = np.transpose(img,(2, 0, 1))
        x[0,:,:,:] = img[0:output_z,0:output_x,0:output_y]
        x = x.astype('float32')

        y = np.zeros((num_labels, output_z, output_x, output_y))
        seg_file = self.seg_files[index]
        seg_3d = nib.load(seg_file)
        seg = seg_3d.get_data()
        seg = np.transpose(seg,(2, 0, 1))
        y[0,:,:,:] = np.ones([output_z,output_x,output_y])
        for i in range(1,num_labels):
            seg_one = seg == labels[i]
            y[i,:,:,:] = seg_one[0:output_z,0:output_x,0:output_y]
            y[0,:,:,:] = y[0,:,:,:] - y[i,:,:,:]
        y = y.astype('float32')
```

Train

```
for batch_idx, (data, target, sub_name) in tqdm.tqdm(
    enumerate(self.train_loader), total=len(self.train_loader),
        desc='Train epoch=%d' % self.epoch, ncols=80, leave=False):

    if self.cuda:
        data, target = data.cuda(), target.cuda()
    data, target = Variable(data), Variable(target)

    pred = self.model(data)
    self.optim.zero_grad()
    loss = dice_loss_3d(pred*100 ,target)
    loss.backward()
    self.optim.step()
```

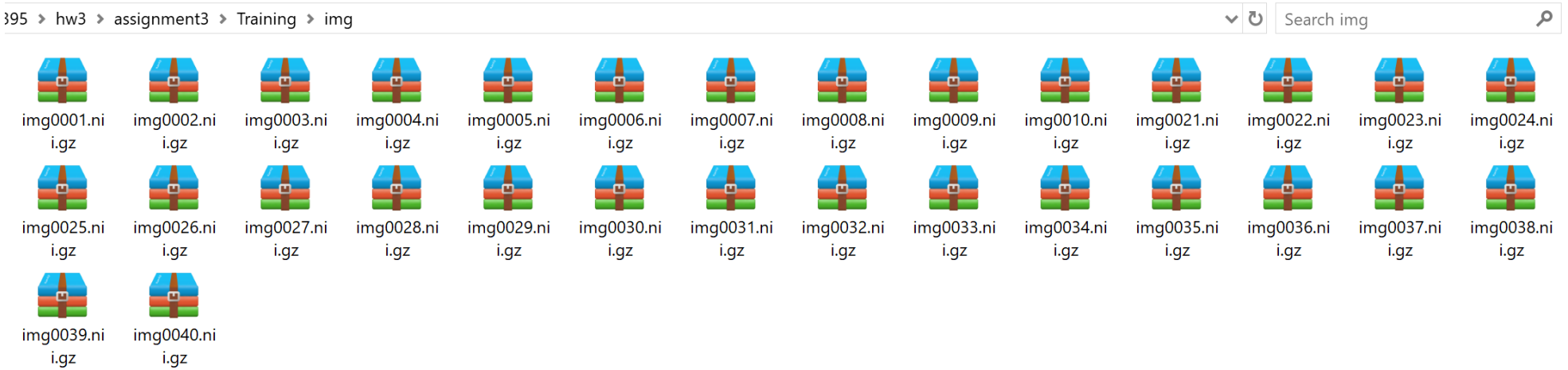
Test

```
for batch_idx, (data,target,sub_name) in tqdm.tqdm(
        # enumerate(self.test_loader), total=len(self.test_loader),
        enumerate(self.test_loader), total=len(self.test_loader),
        desc='Valid epoch=%d' % self.epoch, ncols=80,
        leave=False):

    if self.cuda:
        data, target = data.cuda(), target.cuda()
    data, target = Variable(data,volatile=True), Variable(target,volatile=True)

    pred = self.model(data)
    lbl_pred = pred.data.max(1)[1].cpu().numpy()[:,:, :].astype('uint8')
```

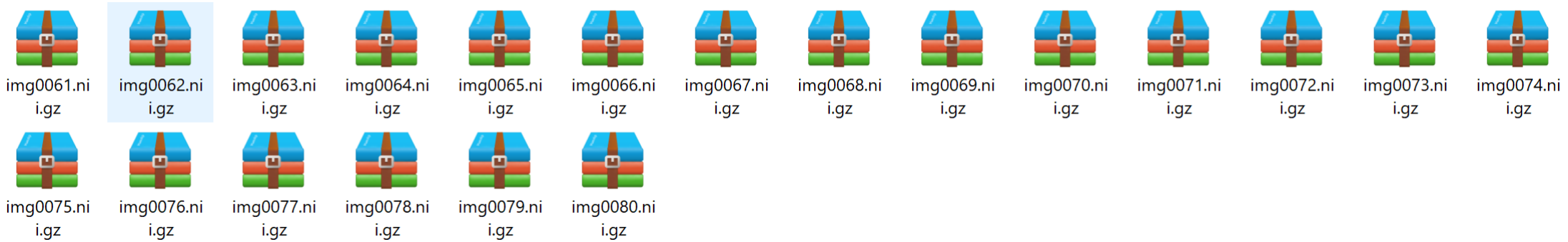
Save Image

```
batch_num = lbl_pred.shape[0]
for si in range(batch_num):
    curr_sub_name = sub_name[si]
    out_img_dir = os.path.join(results_epoch_dir, 'seg')
    mkdir(out_img_dir)
    out_nii_file = os.path.join(out_img_dir,('%s_seg.nii.gz'%(curr_sub_name)))
    img = nib.load(img_file)
    seg_img = nib.Nifti1Image(lbl_pred[si], img.affine, img.header)
    nib.save(seg_img, out_nii_file)
```

Evaluation


395 › hw3 › assignment3 › Training › img
Search img
img0001.nii.gz
img0002.nii.gz
img0003.nii.gz
img0004.nii.gz
img0005.nii.gz
img0006.nii.gz
img0007.nii.gz
img0008.nii.gz
img0009.nii.gz
img0010.nii.gz
img0021.nii.gz
img0022.nii.gz
img0023.nii.gz
img0024.nii.gz
img0025.nii.gz
img0026.nii.gz
img0027.nii.gz
img0028.nii.gz
img0029.nii.gz
img0030.nii.gz
img0031.nii.gz
img0032.nii.gz
img0033.nii.gz
img0034.nii.gz
img0035.nii.gz
img0036.nii.gz
img0037.nii.gz
img0038.nii.gz
img0039.nii.gz
img0040.nii.gz
-8395 › hw3 › assignment3 › Training › label
Search label
label0001.nii.gz
label0002.nii.gz
label0003.nii.gz
label0004.nii.gz
label0005.nii.gz
label0006.nii.gz
label0007.nii.gz
label0008.nii.gz
label0009.nii.gz
label0010.nii.gz
label0021.nii.gz
label0022.nii.gz
label0023.nii.gz
label0024.nii.gz
label0025.nii.gz
label0026.nii.gz
label0027.nii.gz
label0028.nii.gz
label0029.nii.gz
label0030.nii.gz
label0031.nii.gz
label0032.nii.gz
label0033.nii.gz
label0034.nii.gz
label0035.nii.gz
label0036.nii.gz
label0037.nii.gz
label0038.nii.gz
label0039.nii.gz
label0040.nii.gz

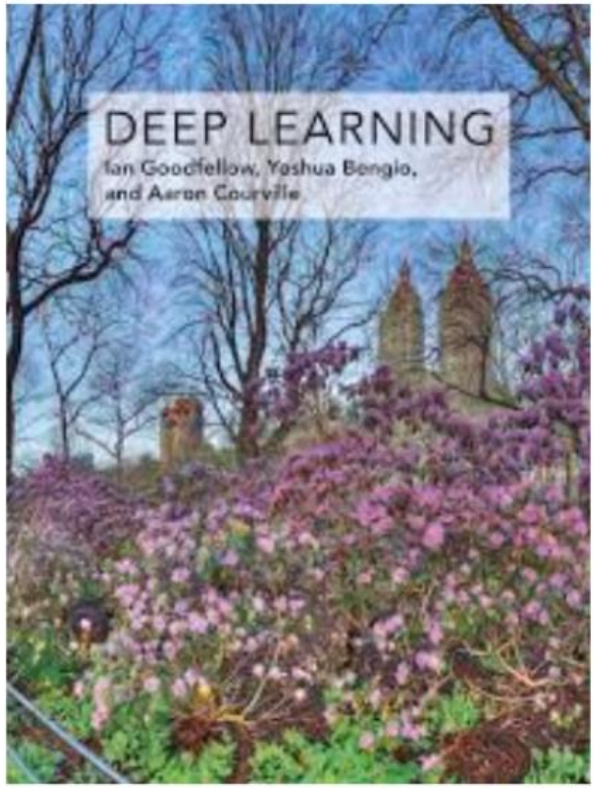
Evaluation


img0061.nii.gz
img0062.nii.gz
img0063.nii.gz
img0064.nii.gz
img0065.nii.gz
img0066.nii.gz
img0067.nii.gz
img0068.nii.gz
img0069.nii.gz
img0070.nii.gz
img0071.nii.gz
img0072.nii.gz
img0073.nii.gz
img0074.nii.gz
img0075.nii.gz
img0076.nii.gz
img0077.nii.gz
img0078.nii.gz
img0079.nii.gz
img0080.nii.gz

Attention

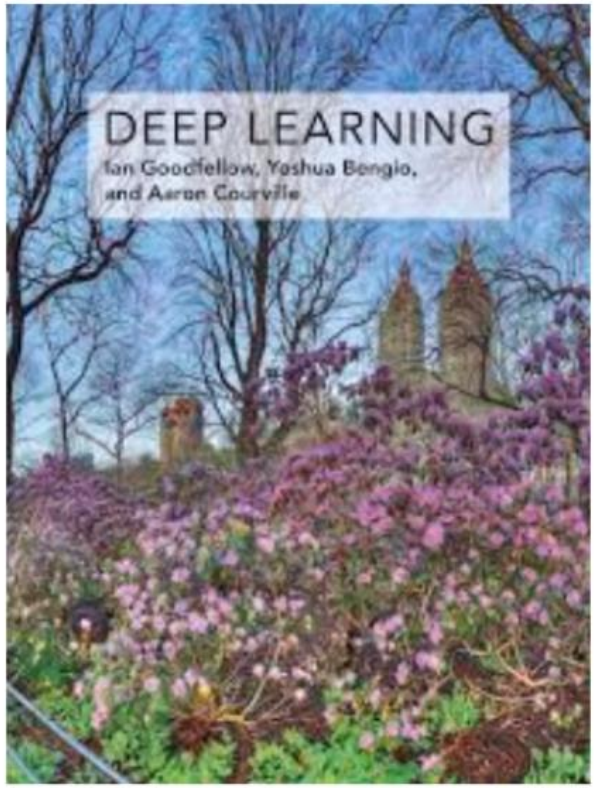
https://nndl.github.io/

Visual Attention



How is convolution equivariant w.r.t. translation ? https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention



Deep Learning

How is convolution equivariant w.r.t. translation ? https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention

Chapter 9

Convolutional Networks

Convolutional networks (LeCun, 1989), also known as **convolutional neural networks**, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called **convolution**. Convolution is a specialized kind of linear operation. *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

In this chapter, we first describe what convolution is. Next, we explain the motivation behind using convolution in a neural network. We then describe an operation called **pooling**, which almost all convolutional networks employ. Usually, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics. We describe several variants on the convolution function that are widely used in practice for neural networks. We also show how convolution may be applied to many kinds of data, with different numbers of dimensions. We

https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention

Chapter 9

Convolutional Networks

Convolutional networks (LeCun, 1989), also known as **convolutional neural networks**, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called **convolution**. Convolution is a specialized kind of linear operation. *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

In this chapter, we first describe what convolution is. Next, we explain the motivation behind using convolution in a neural network. We then describe an operation called **pooling**, which almost all convolutional networks employ. Usually, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics. We describe several variants on the convolution function that are widely used in practice for neural networks. We also show how convolution may be applied to many kinds of data, with different numbers of dimensions. We

each pixel in the original image and subtracting the value of its neighboring pixel on the left. This shows the strength of all the vertically oriented edges in the input image, which can be a useful operation for object detection. Both images are 280 pixels tall. The input image is 320 pixels wide, while the output image is 319 pixels wide. This transformation can be described by a convolution kernel containing two elements, and requires $319 \times 280 \times 3 = 267,960$ floating-point operations (two multiplications and one addition per output pixel) to compute using convolution. To describe the same transformation with a matrix multiplication would take $320 \times 280 \times 319 \times 280$, or over eight billion, entries in the matrix, making convolution four billion times more efficient for representing this transformation. The straightforward matrix multiplication algorithm performs over sixteen billion floating point operations, making convolution roughly 60,000 times more efficient computationally. Of course, most of the entries of the matrix would be zero. If we stored only the nonzero entries of the matrix, then both matrix multiplication and convolution would require the same number of floating-point operations to compute. The matrix would still need to contain $2 \times 319 \times 280 = 178,640$ entries. Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small local region across the entire input. Photo credit: Paula Goodfellow.

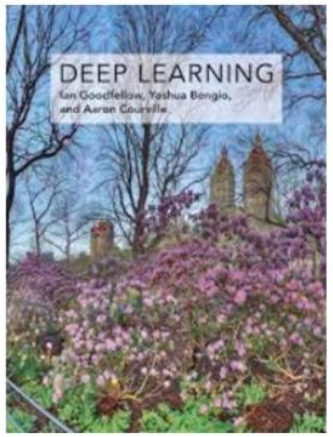
efficiency of a linear function for detecting edges in an image.

In the case of convolution, the particular form of parameter sharing causes the layer to have a property called **equivariance** to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. In the case of convolution, if we let g be any function that translates the input, that is, shifts it, then the convolution function is equivariant to g. For example, let I be a function giving image brightness at integer coordinates. Let g be a function mapping one image function to another image function, such that $I' = g(I)$ is the image function with $I'(x, y) = I(x - 1, y)$. This shifts every pixel of I one unit to the right. If we apply this transformation to I, then apply convolution, the result will be the same as if we applied convolution to I', then applied the transformation

334

https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention



→ 1 Week

Chapter 9

Convolutional Networks

Convolutional networks (LeCun, 1989), also known as **convolutional neural networks**, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called **convolution**. Convolution is a specialized kind of linear operation. *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*

In this chapter, we first describe what convolution is. Next, we explain the motivation behind using convolution in a neural network. We then describe an operation called **pooling**, which almost all convolutional networks employ. Usually, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics. We describe several variants on the convolution function that are widely used in practice for neural networks. We also show how convolution may be applied to many kinds of data, with different numbers of dimensions. We then discuss means of making convolution more efficient. Convolutional networks stand out as an example of neuroscientific principles influencing deep learning. We discuss these neuroscientific principles, then conclude with comments about the role convolutional networks have played in the history of deep learning. One topic this chapter does not address is how to choose the architecture of your convolutional network. The goal of this chapter is to describe the kinds of tools that convolutional networks provide, while chapter 11 describes general guidelines

326

→ 1 minute

https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention

zero. If we stored only the nonzero entries of the matrix, then both matrix multiplication and convolution would require the same number of floating-point operations to compute. The matrix would still need to contain $2 \times 319 \times 280 = 178,640$ entries. Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small local region across the entire input. Photo credit: Paula Goodfellow.

efficiency of a linear function for detecting edges in an image.

In the case of convolution, the particular form of parameter sharing causes the layer to have a property called **equivariance** to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. In the case of convolution, if we let g be any function that translates the input, that is, shifts it, then the convolution function is equivariant to g. For example, let I be a function giving image brightness at integer coordinates. Let g be a function mapping one image function to another image function, such that $I' = g(I)$ is the image function with $I'(x, y) = I(x - 1, y)$. This shifts every pixel of I one unit to the right. If we apply this transformation to I, then apply convolution, the result will be the same as if we applied convolution to I', then applied the transformation

334

https://www.youtube.com/watch?v=W2rWgXJBZhU

Visual Attention

zero. If we stored only the nonzero entries of the matrix, then both matrix multiplication and convolution would require the same number of floating-point operations to compute. The matrix would still need to contain $2 \times 319 \times 280 = 178,640$ entries. Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small local region across the entire input. Photo credit: Paula Goodfellow.

efficiency of a linear function for detecting edges in an image.

In the case of convolution, the particular form of parameter sharing causes the layer to have a property called **equivariance** to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. In the case of convolution, if we let g be any function that translates the input, that is, shifts it, then the convolution function is equivariant to g. For example, let I be a function giving image brightness at integer coordinates. Let g be a function mapping one image function to another image function, such that $I' = g(I)$ is the image function with $I'(x, y) = I(x - 1, y)$. This shifts every pixel of I one unit to the right. If we apply this transformation to I, then apply convolution, the result will be the same as if we applied convolution to I', then applied the transformation

334

https://www.youtube.com/watch?v=W2rWgXJBZhU

Attention

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

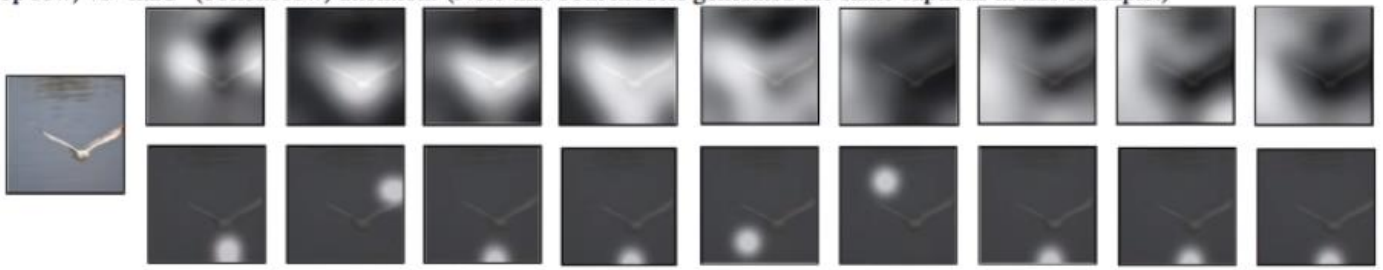


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

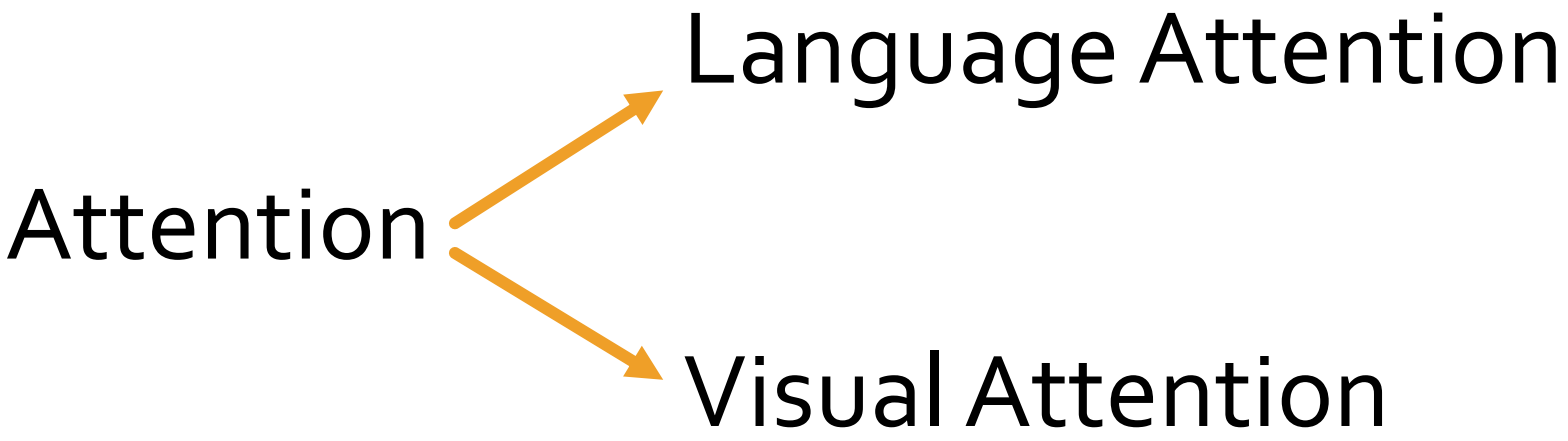
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

https://www.youtube.com/watch?v=W2rWgXJBZhU

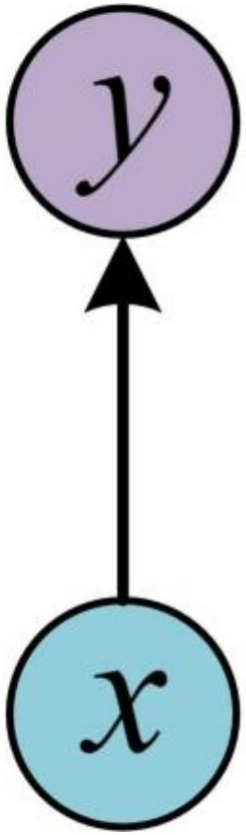
Attentions



Example

I come from Nashville, and there are many beautiful scenes and great bars in Nashville. If you would like to visit _____, I hope to show you around.

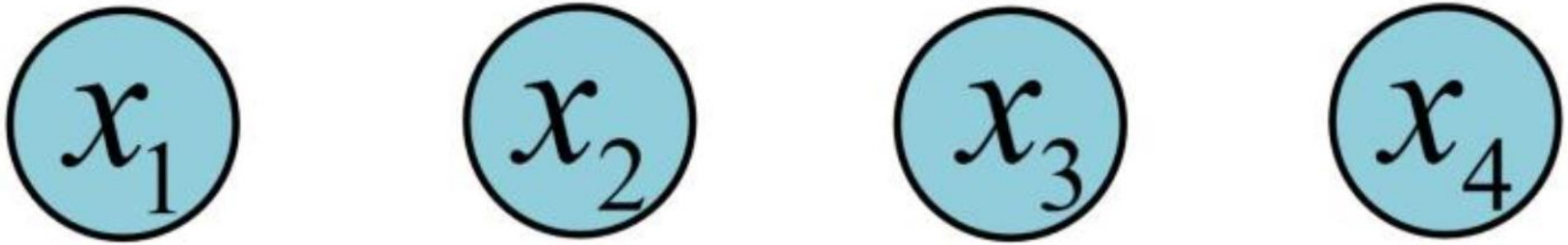
1-to-1



$$y = f(Wx + b)$$

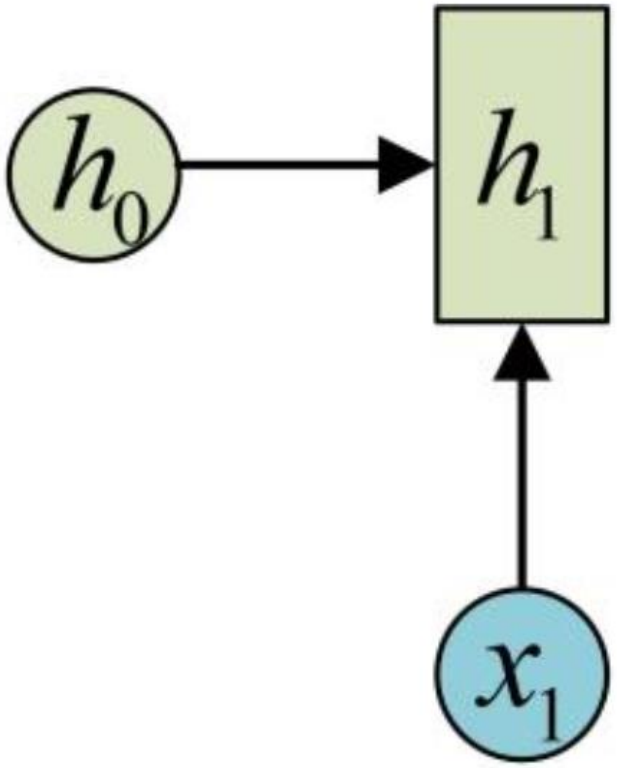
https://zhuanlan.zhihu.com/p/28054589

N-to-1



https://zhuanlan.zhihu.com/p/28054589

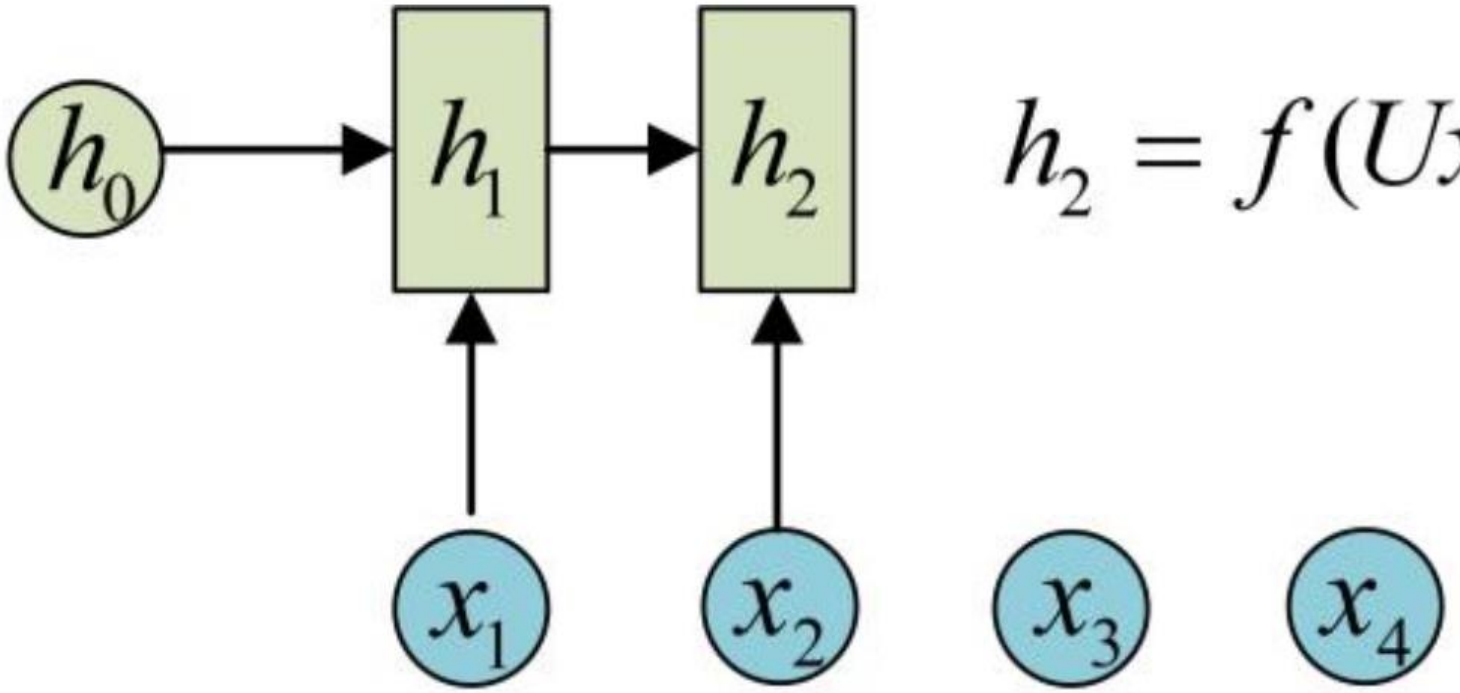
Hidden State



$$h_1 = f(Ux_1 + Wh_0 + b)$$

https://zhuanlan.zhihu.com/p/28054589

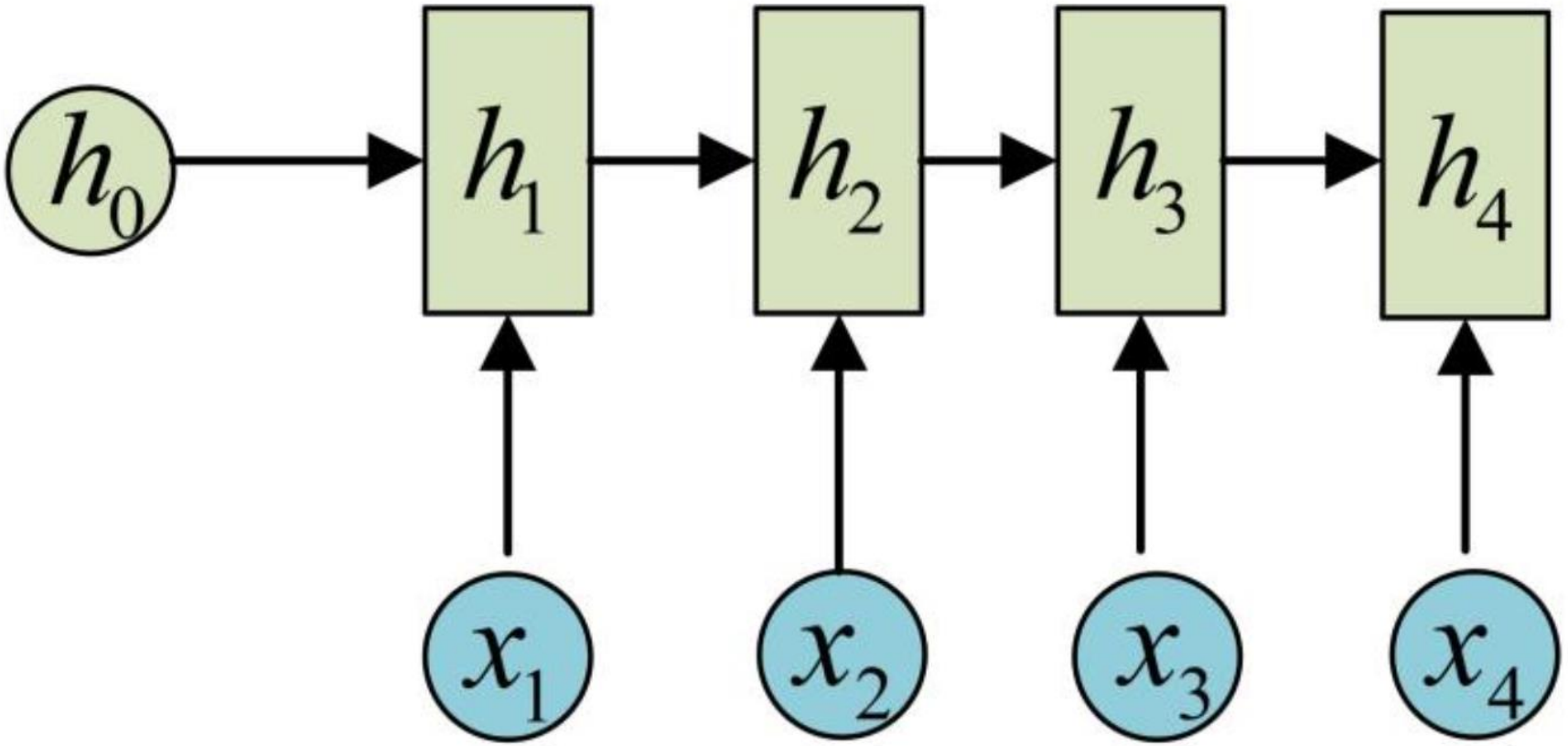
Hidden State



$$h_2 = f(Ux_2 + Wh_1 + b)$$

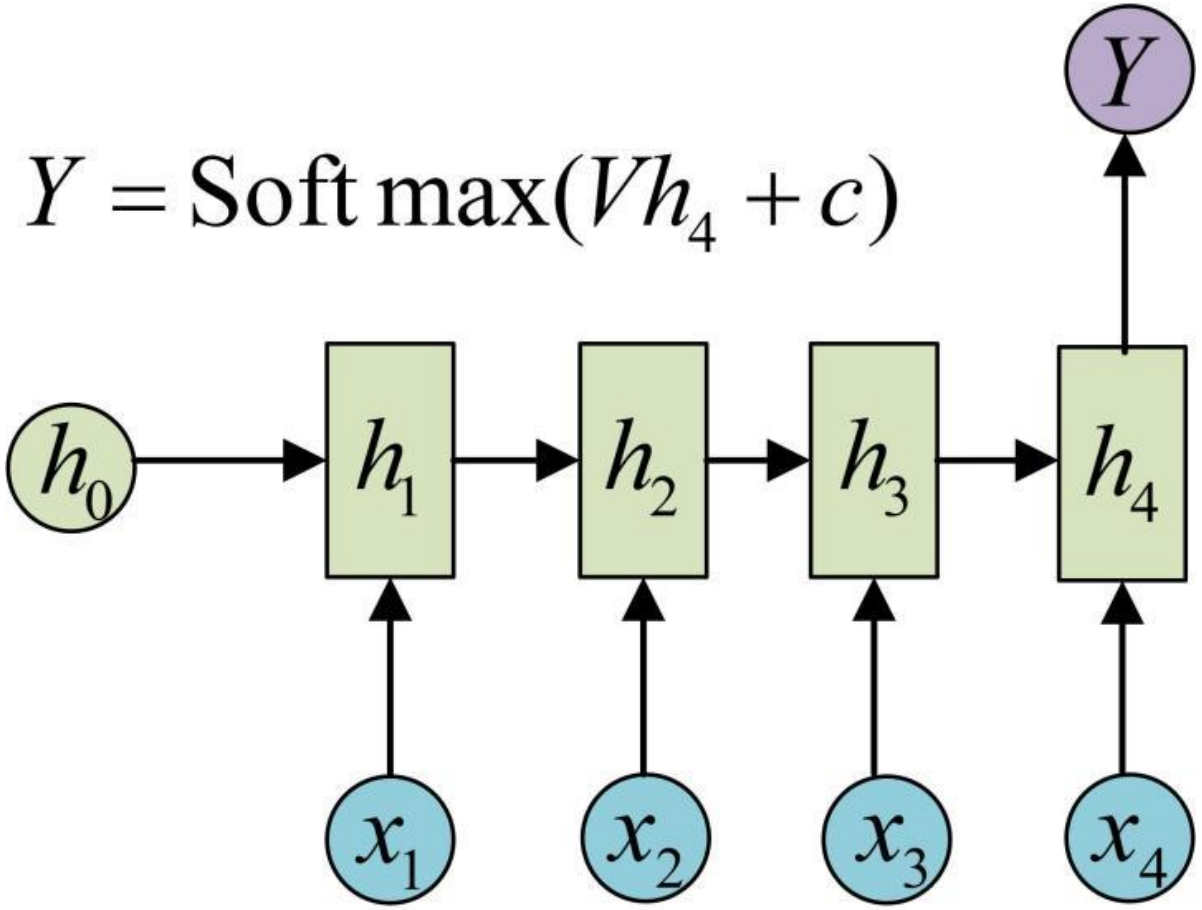
https://zhuanlan.zhihu.com/p/28054589

Hidden State



https://zhuanlan.zhihu.com/p/28054589

N vs 1



https://zhuanlan.zhihu.com/p/28054589

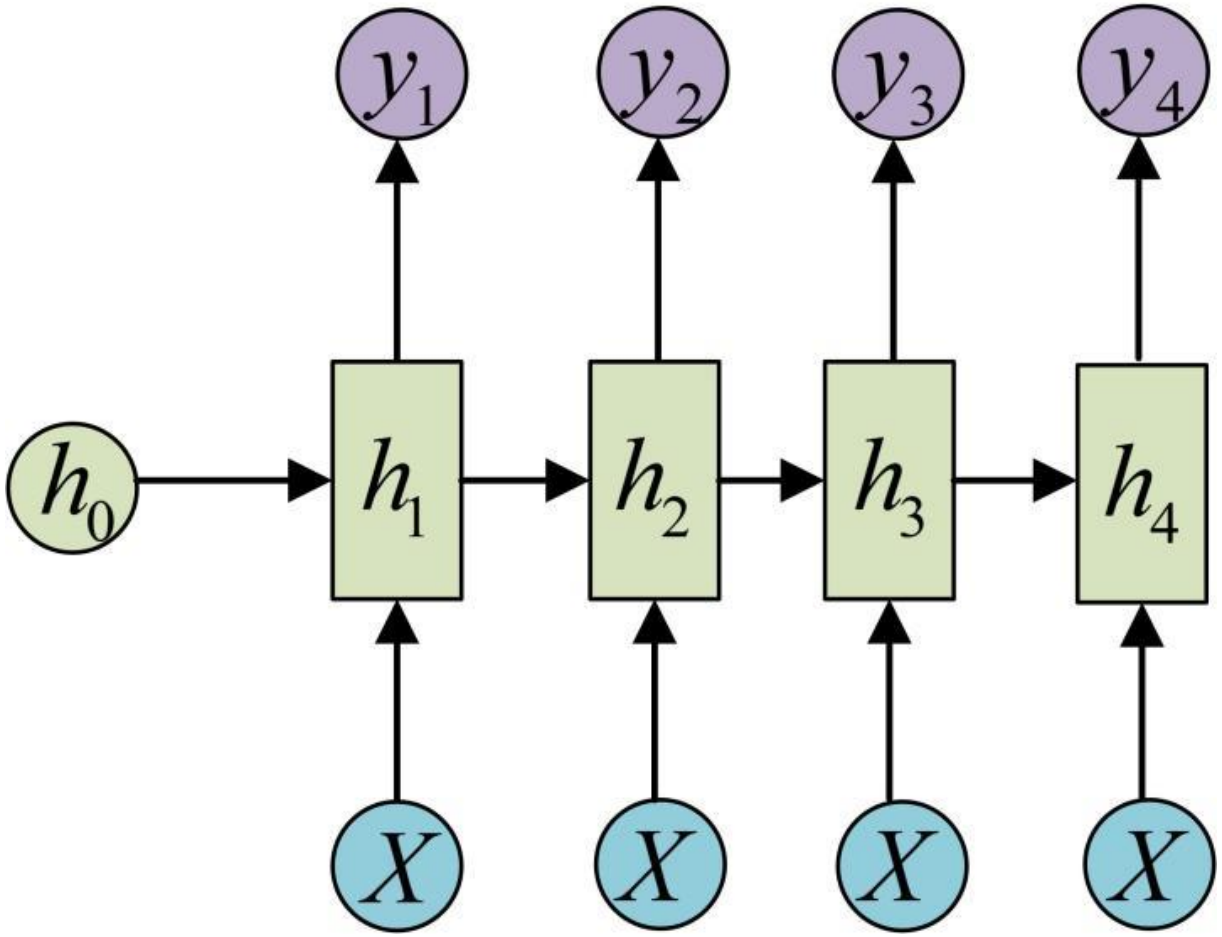
Example

I come from Nashville, and there are many beautiful scenes and great bars in Nashville. If you would like to visit ___Nashville___, I hope to show you around.

N to N, Translation?

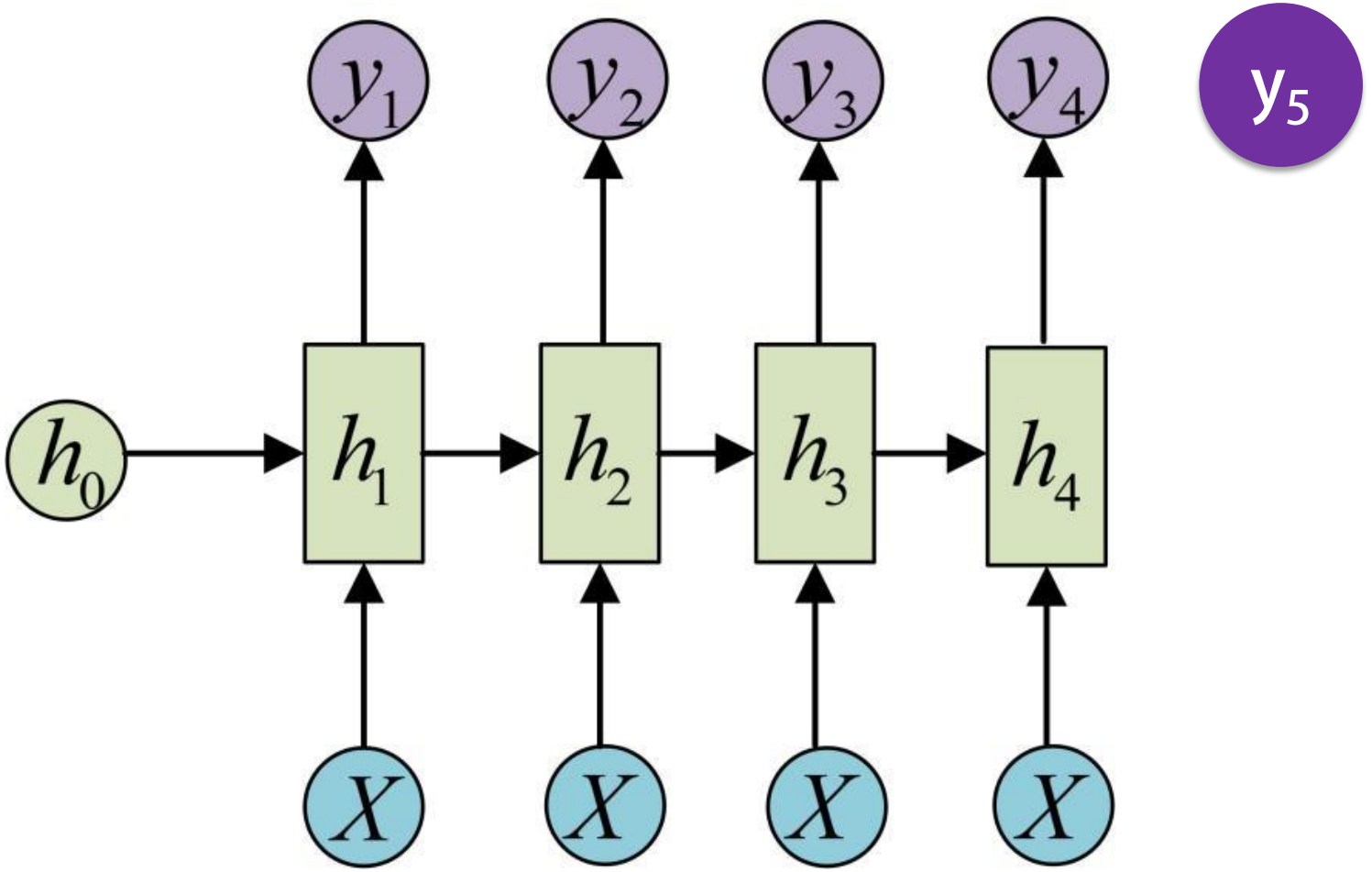
我爱纳村 → I Love Nashville

N vs N



https://zhuanlan.zhihu.com/p/28054589

N vs M?



https://zhuanlan.zhihu.com/p/28054589

Seq2Seq

Sequence to Sequence Learning with Neural Networks

https://arxiv.org › cs ▼ 翻译此页

作者：I Sutskever - 2014 - 被引用次数：4856 - 相关文章

2014年9月10日 - Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this ...

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

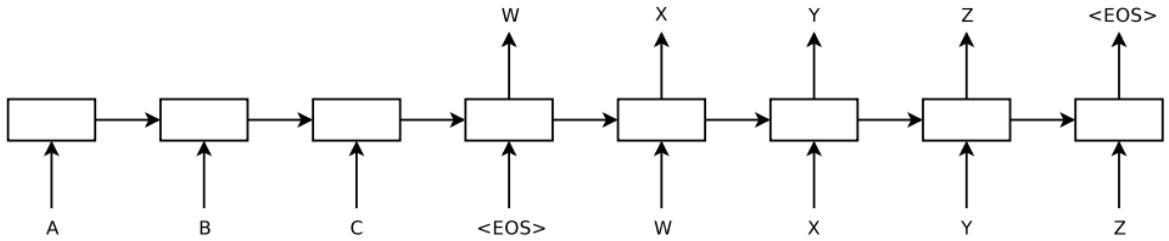
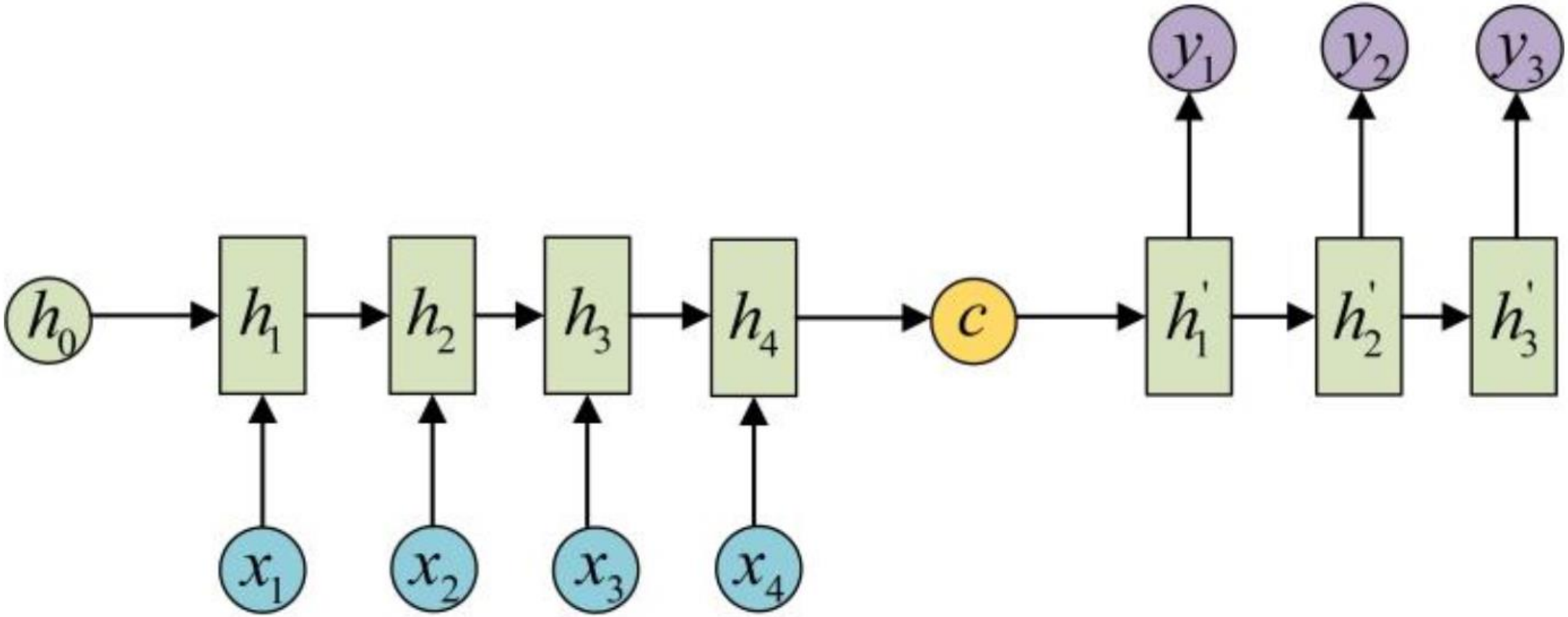


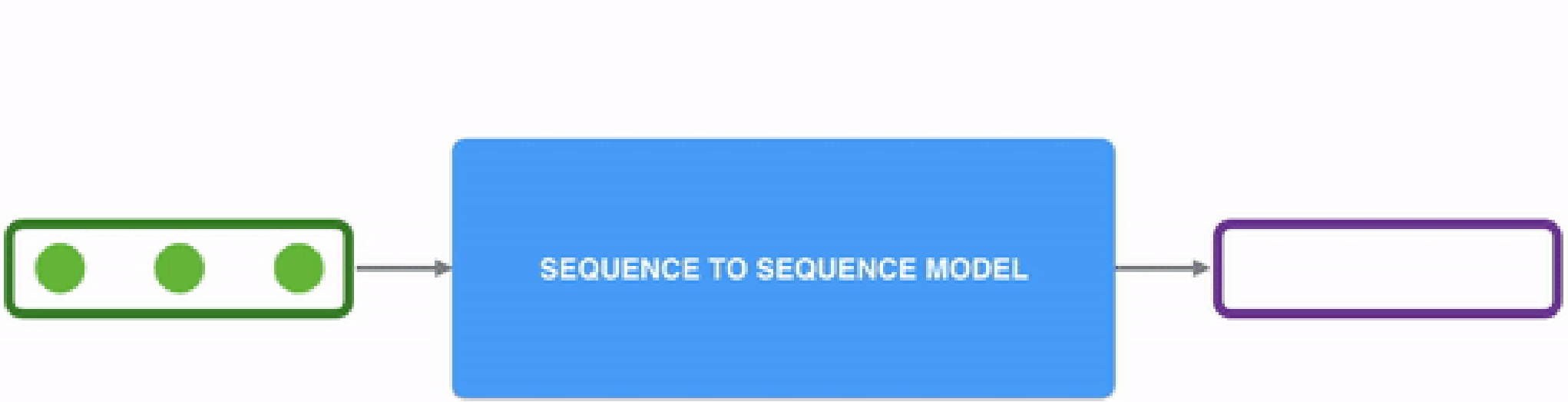
Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Seq2Seq



https://zhuanlan.zhihu.com/p/28054589

Idea



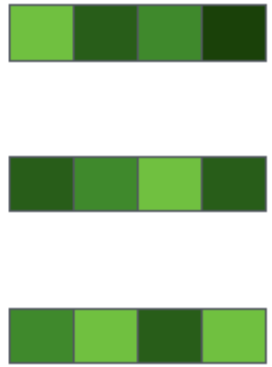
https://towardsdatascience.com/transformers-141e32e69591

Word Embedding

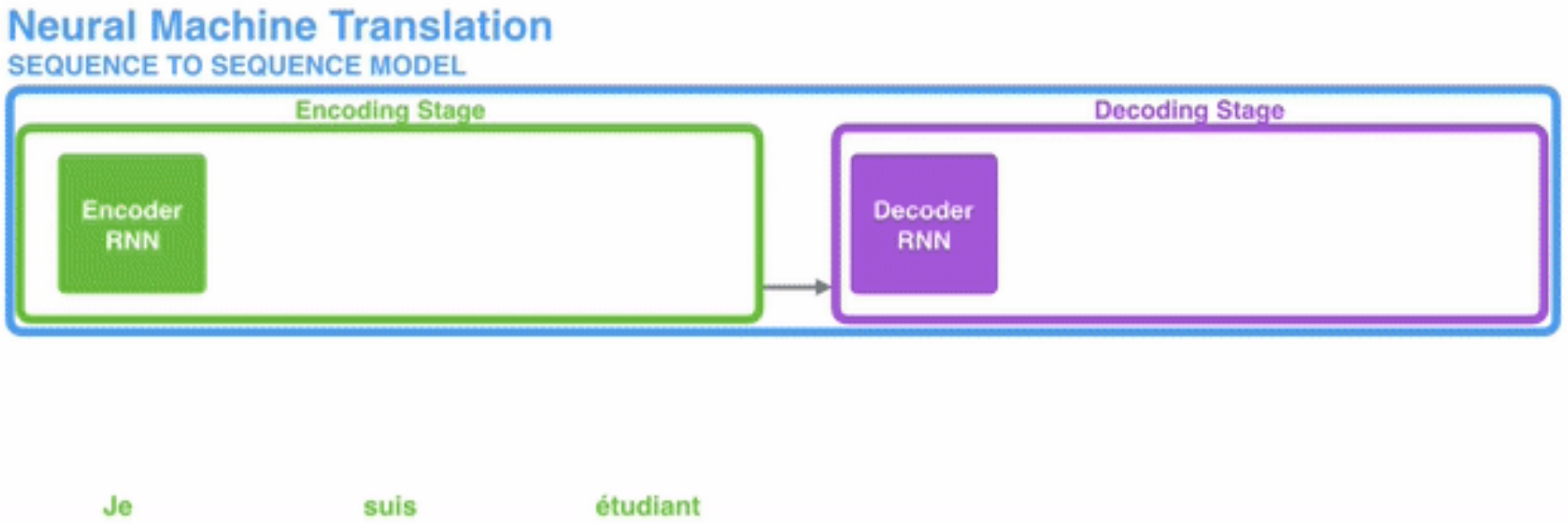
Input

Je
suis
étudiant

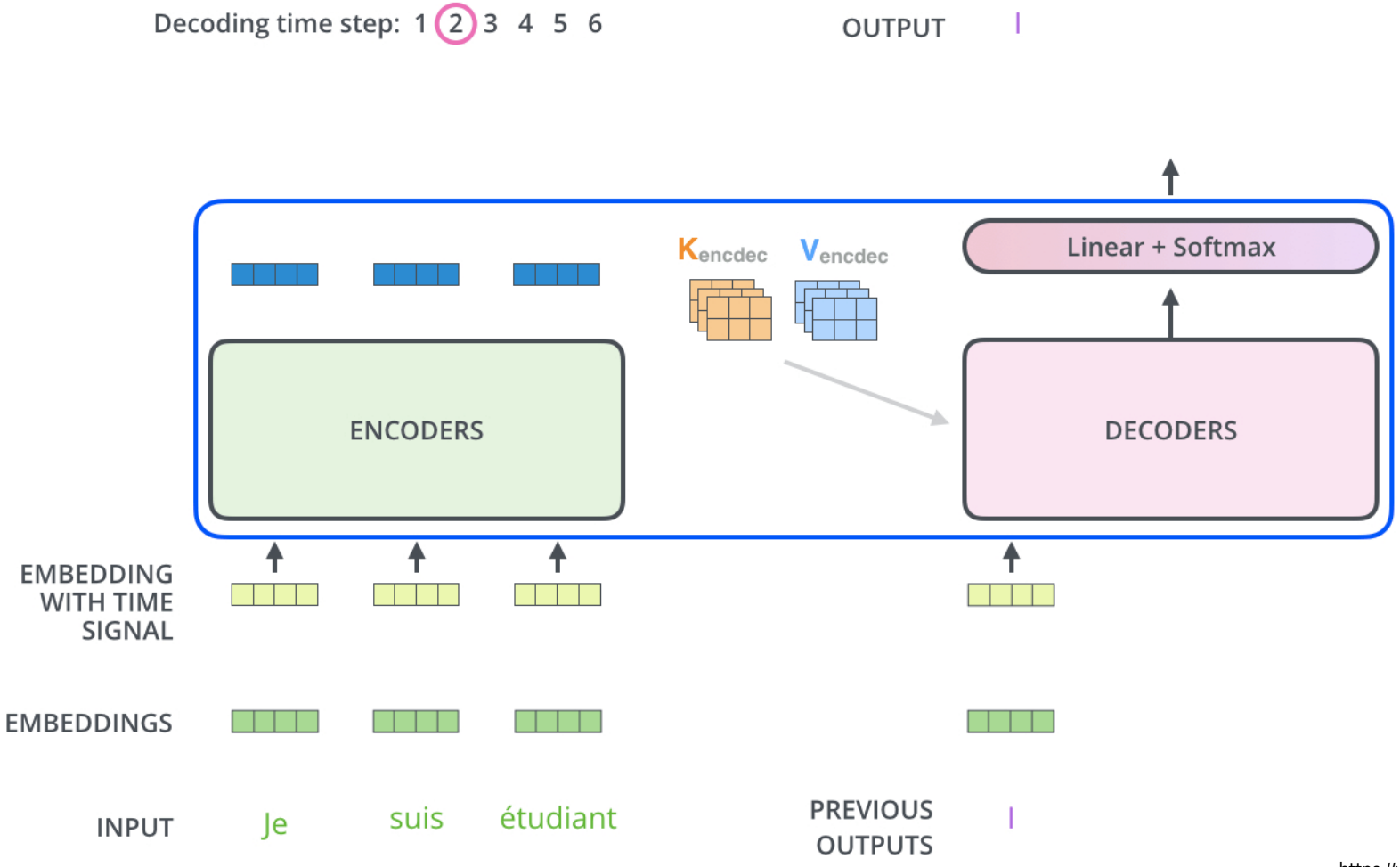
0.901	-0.651	-0.194	-0.822
-0.351	0.123	0.435	-0.200
0.081	0.458	-0.400	0.480



Seq2Seq

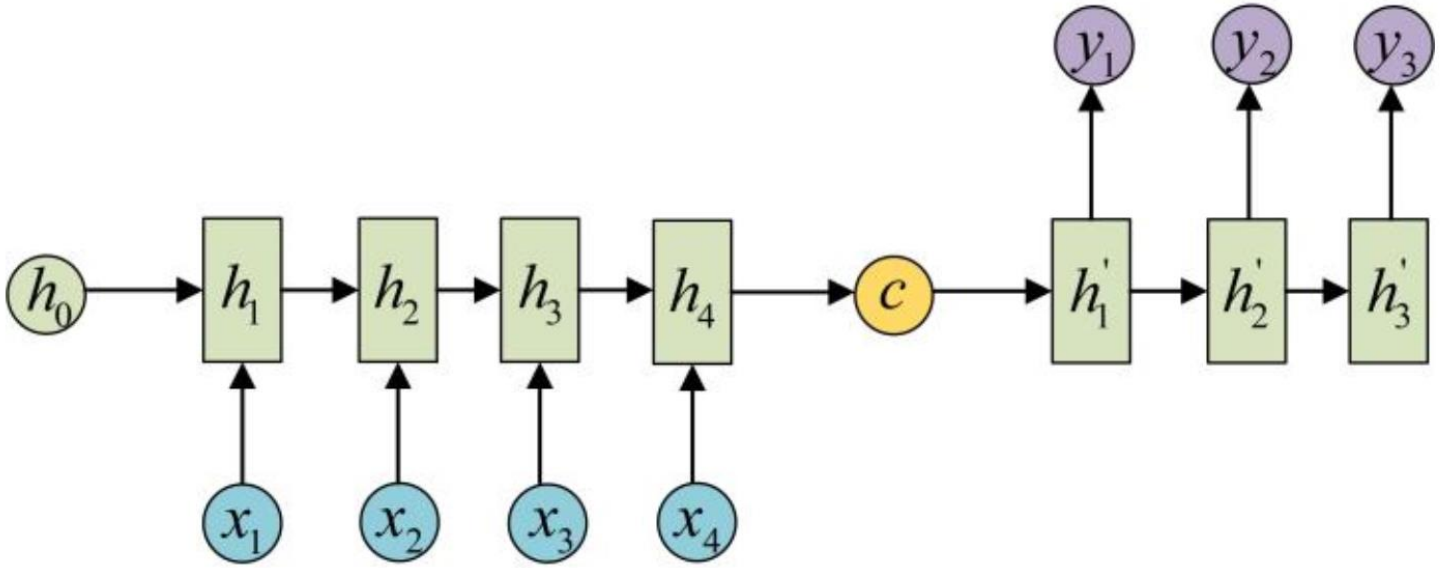

Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL
Encoding Stage
Decoding Stage
Encoder RNN
Decoder RNN
Je
suis
étudiant

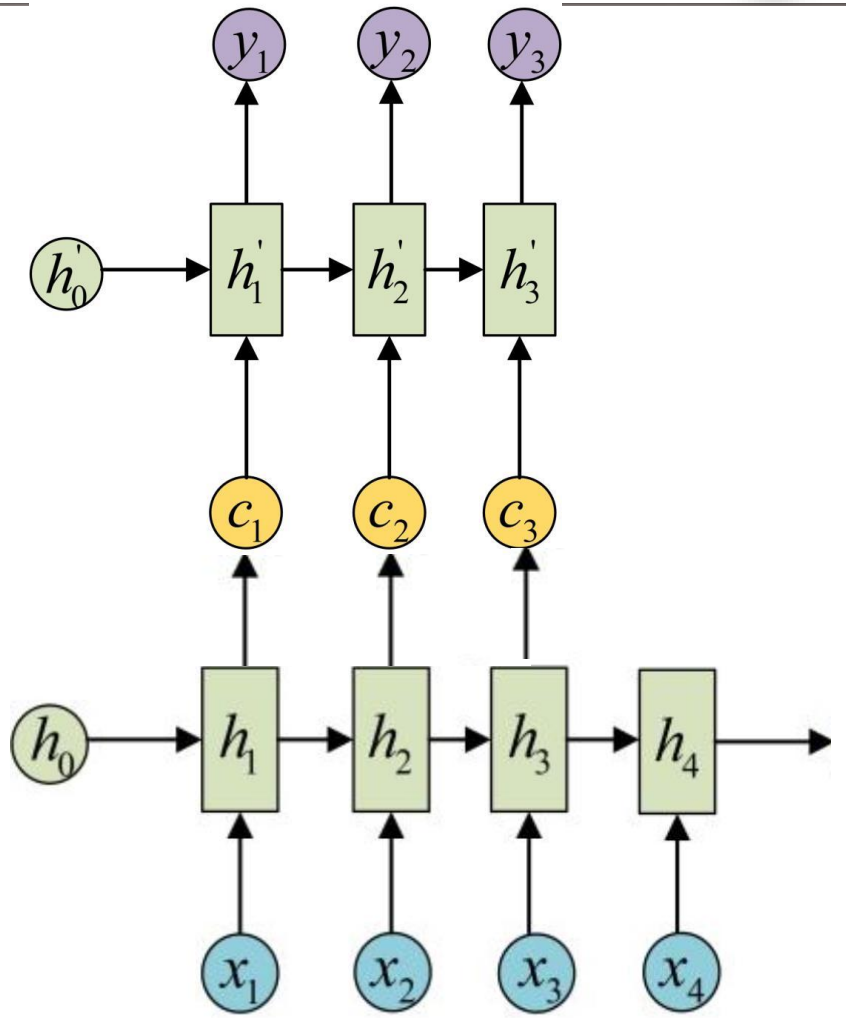
Translation



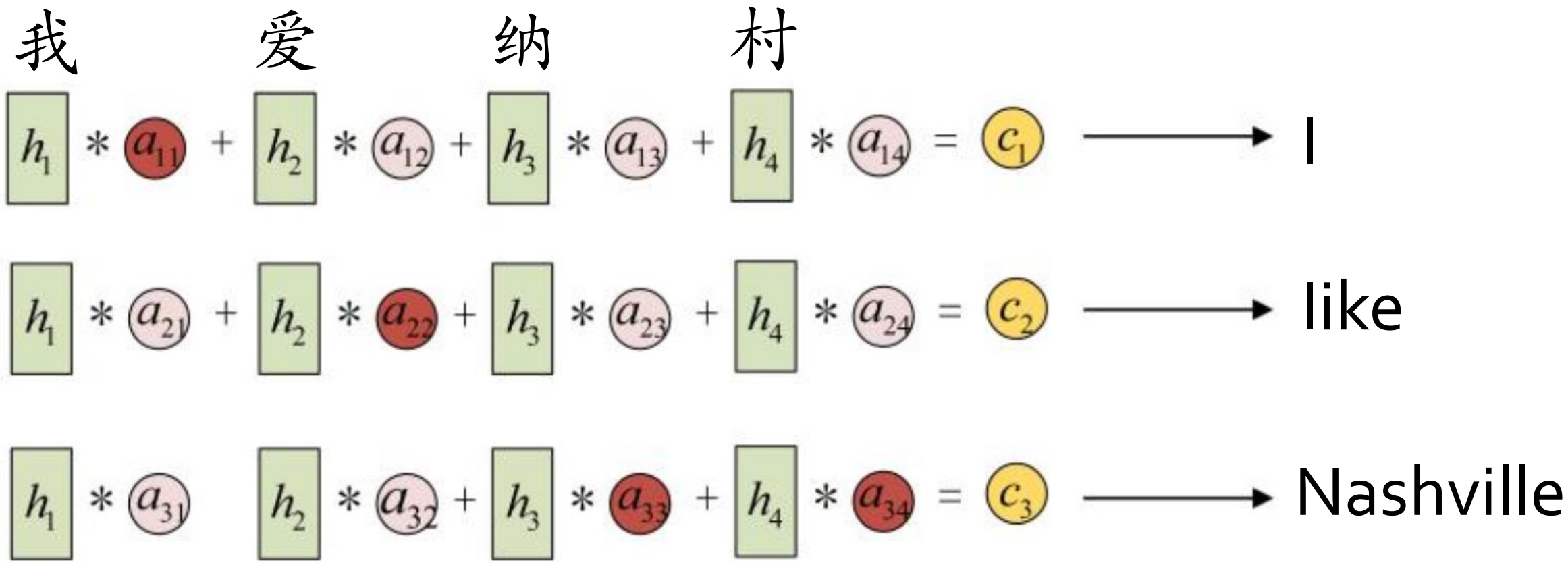
https://ychai.uk/notes/2019/10/17/NN/Transformer-variants-a-peek/

Attention

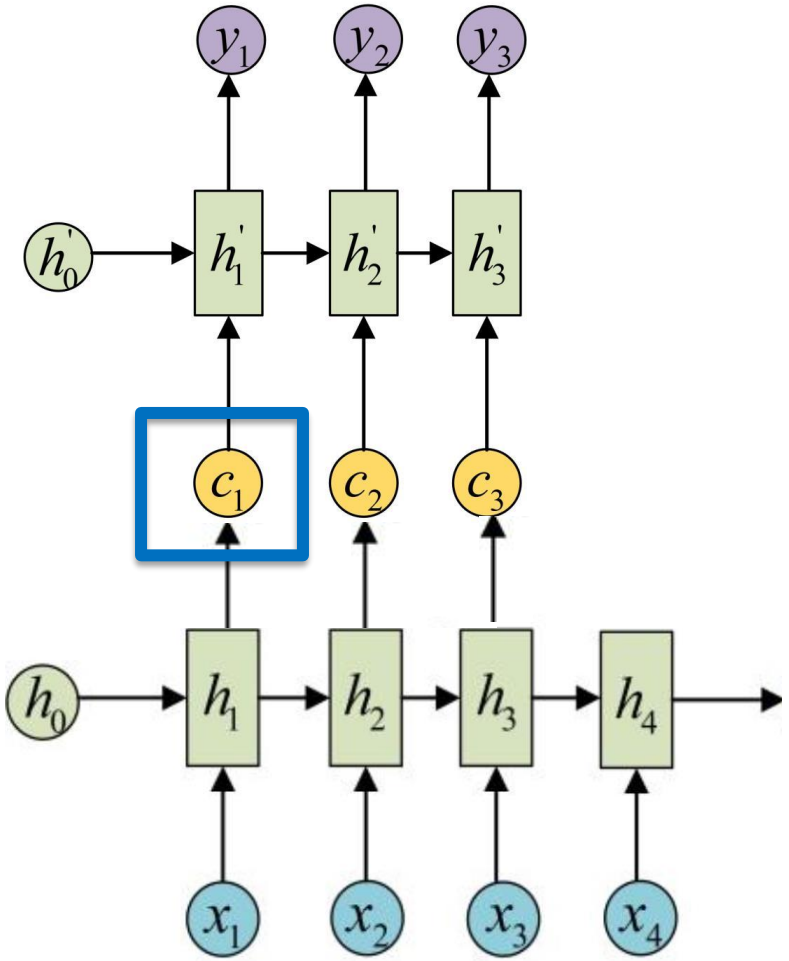

y_1 y_2 y_3
h_0 h_1 h_2 h_3 h_4 c h'_1 h'_2 h'_3
x_1 x_2 x_3 x_4

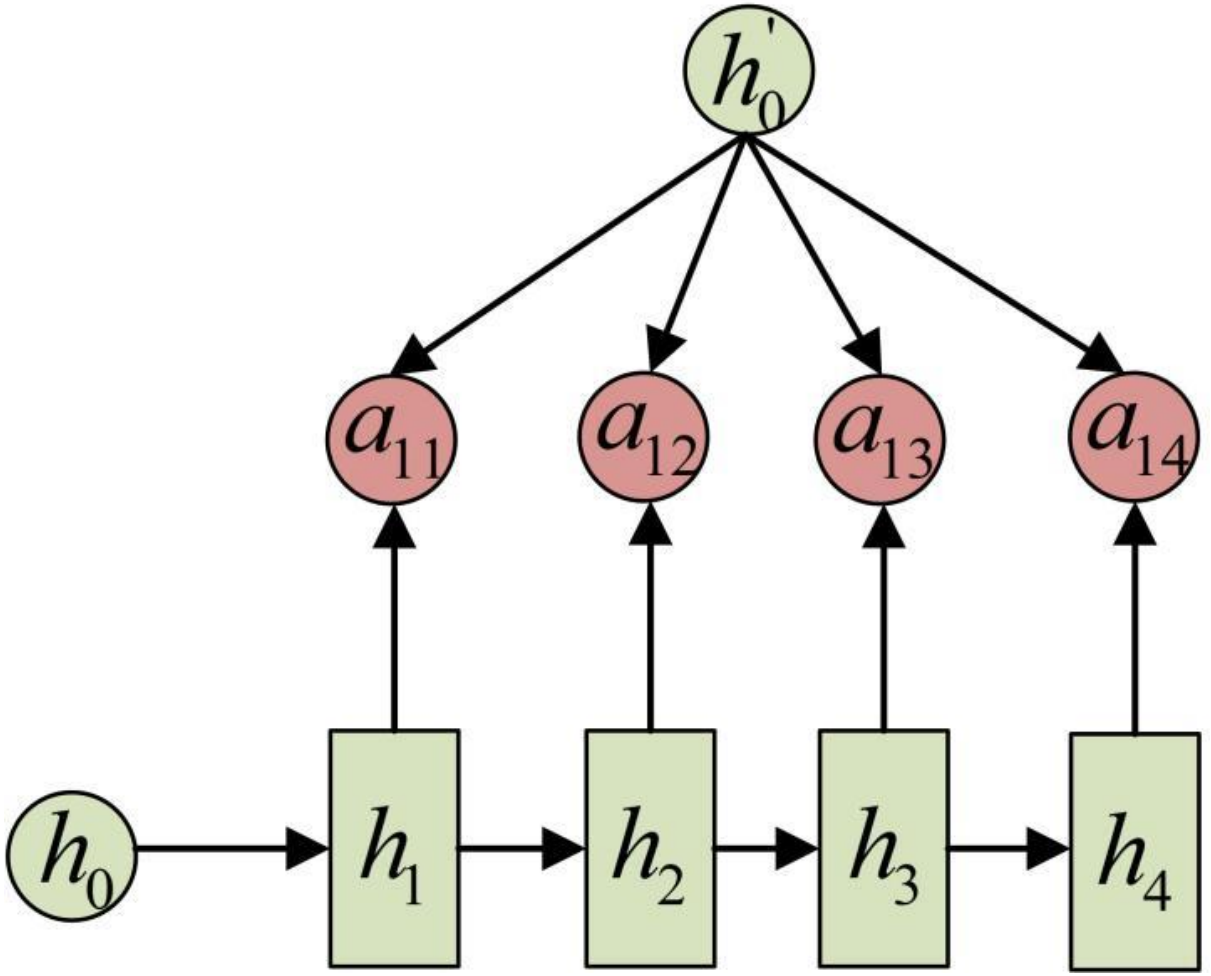

y_1 y_2 y_3
h'_0 h'_1 h'_2 h'_3
c_1 c_2 c_3
h_0 h_1 h_2 h_3 h_4
x_1 x_2 x_3 x_4

Represent Hidden State

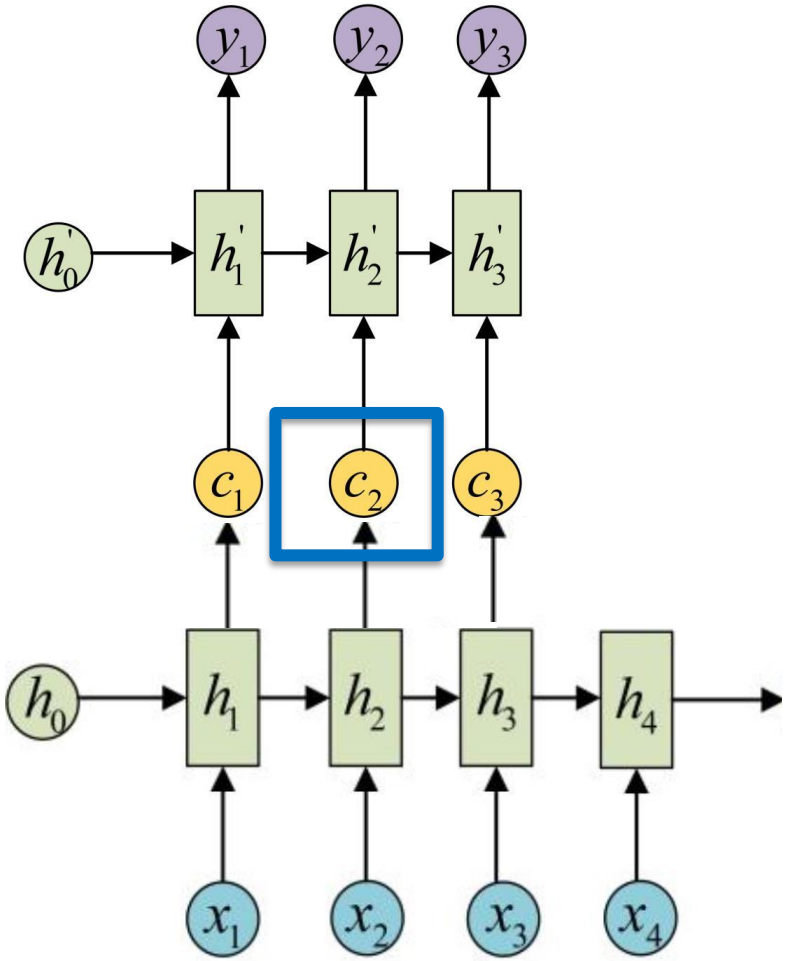


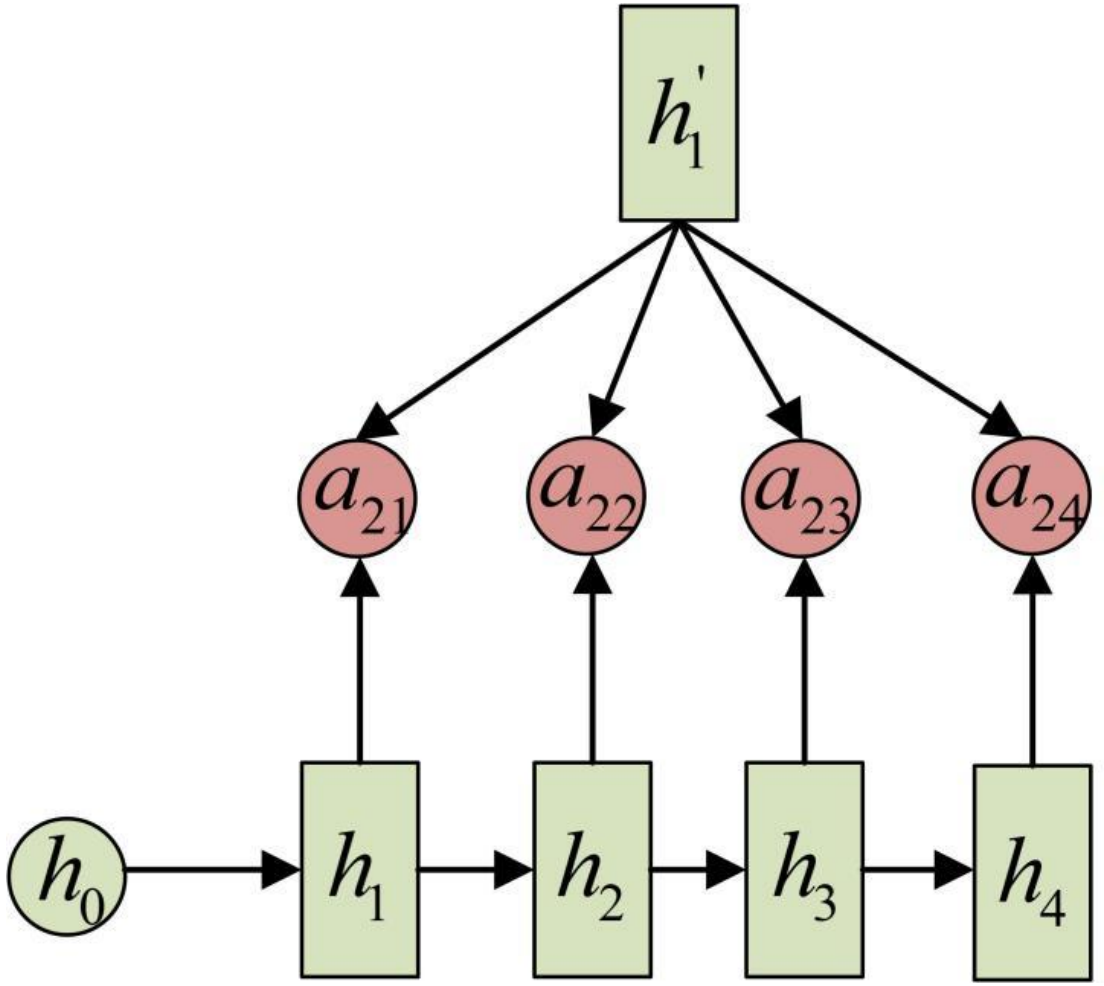
Get Probability



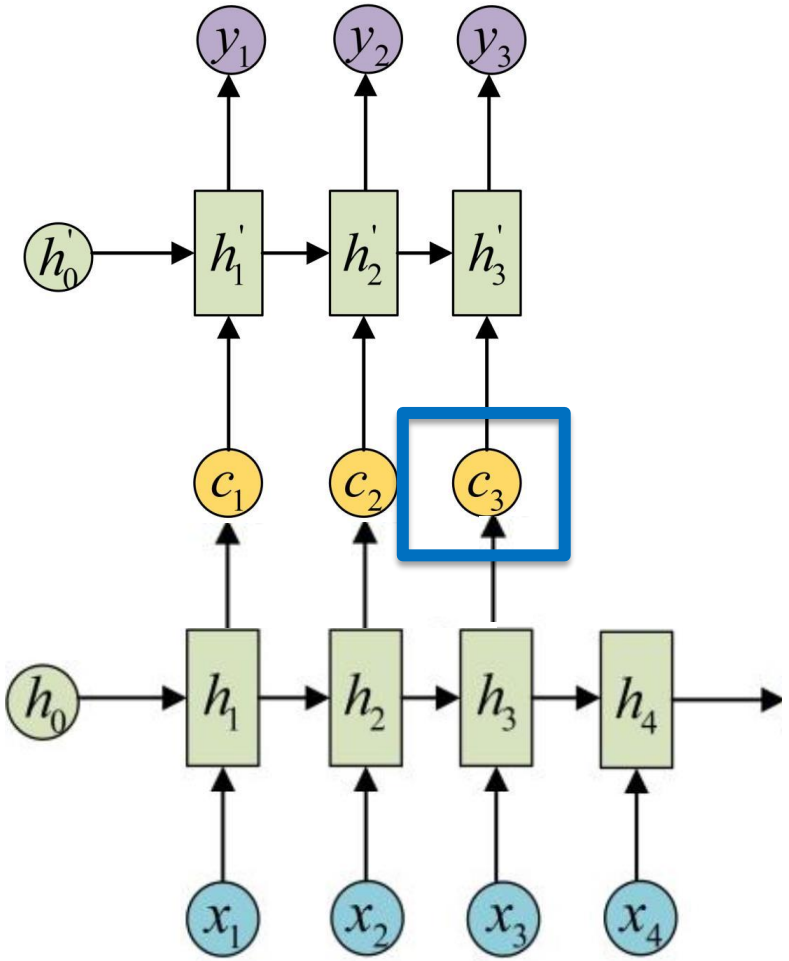


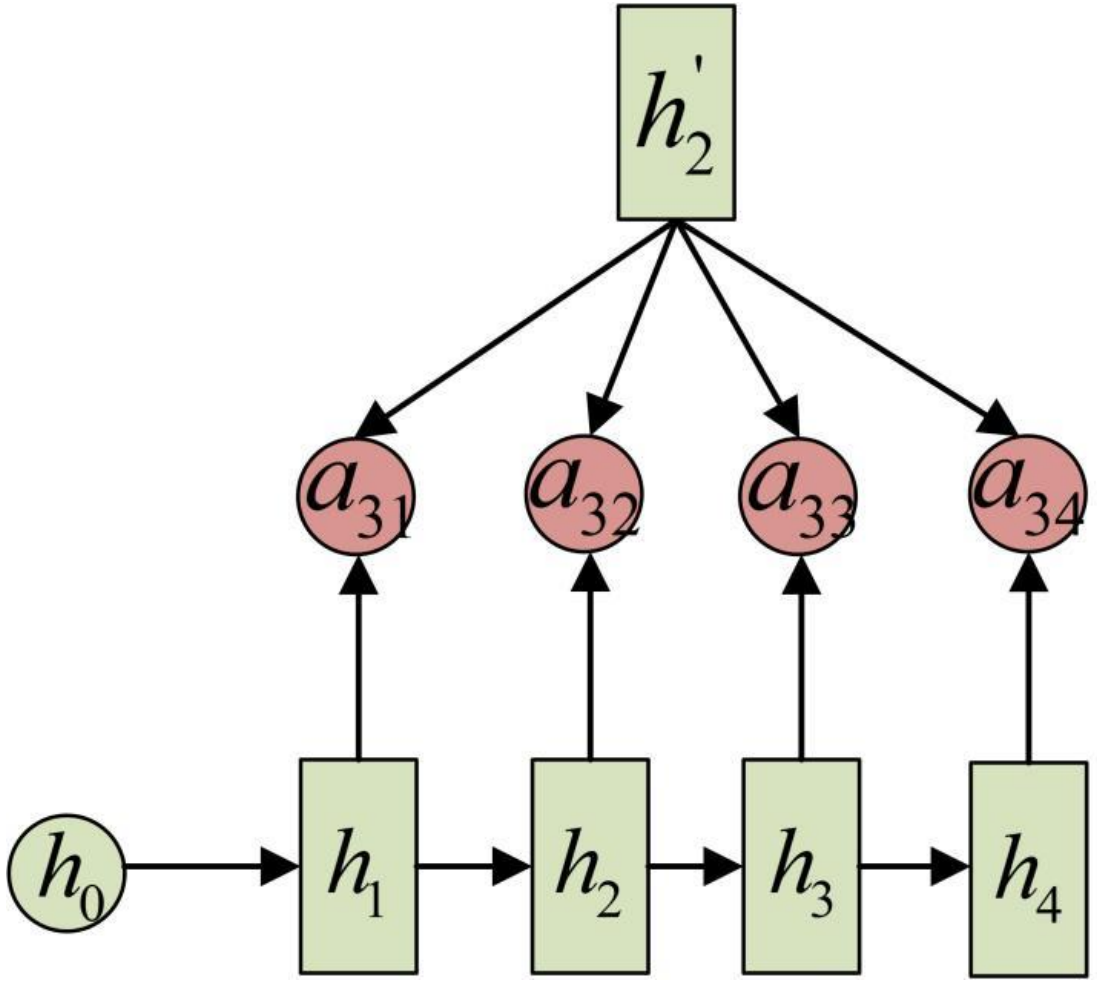
Get Probability



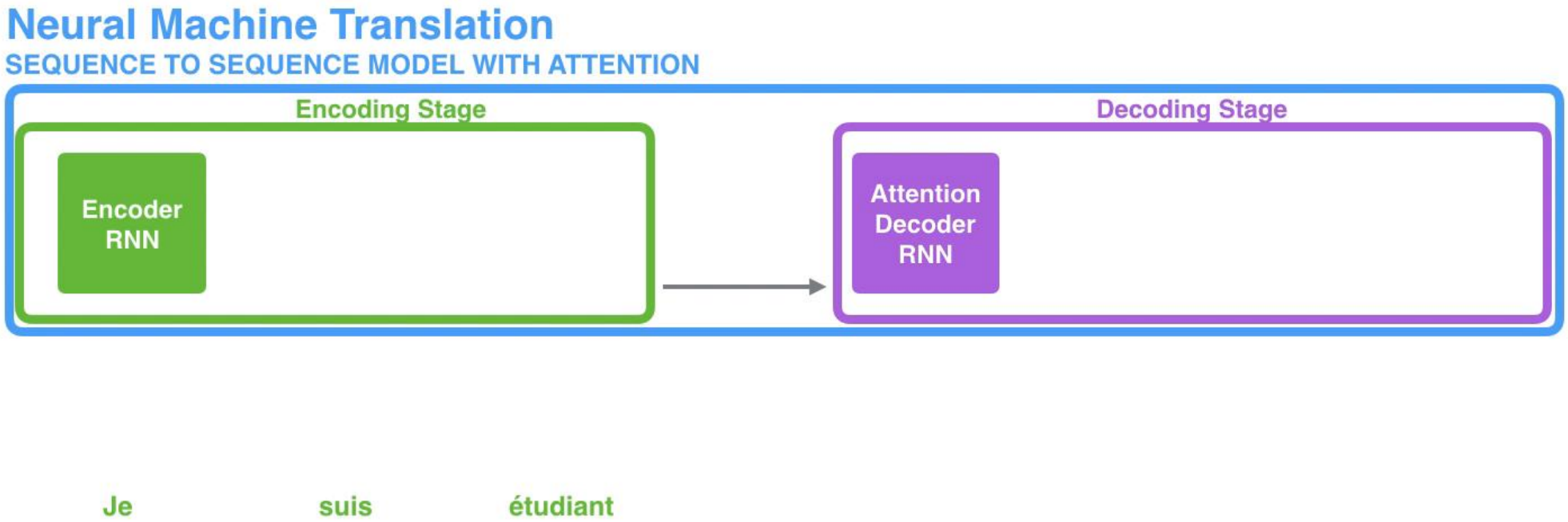


Get Probability



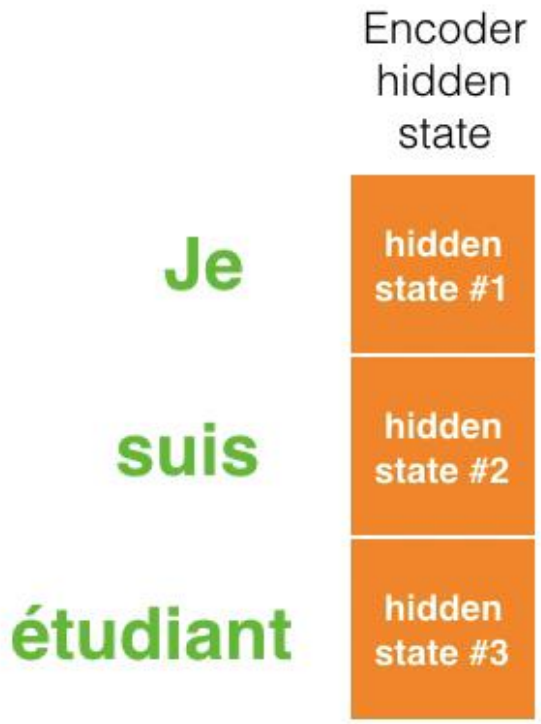


Translation



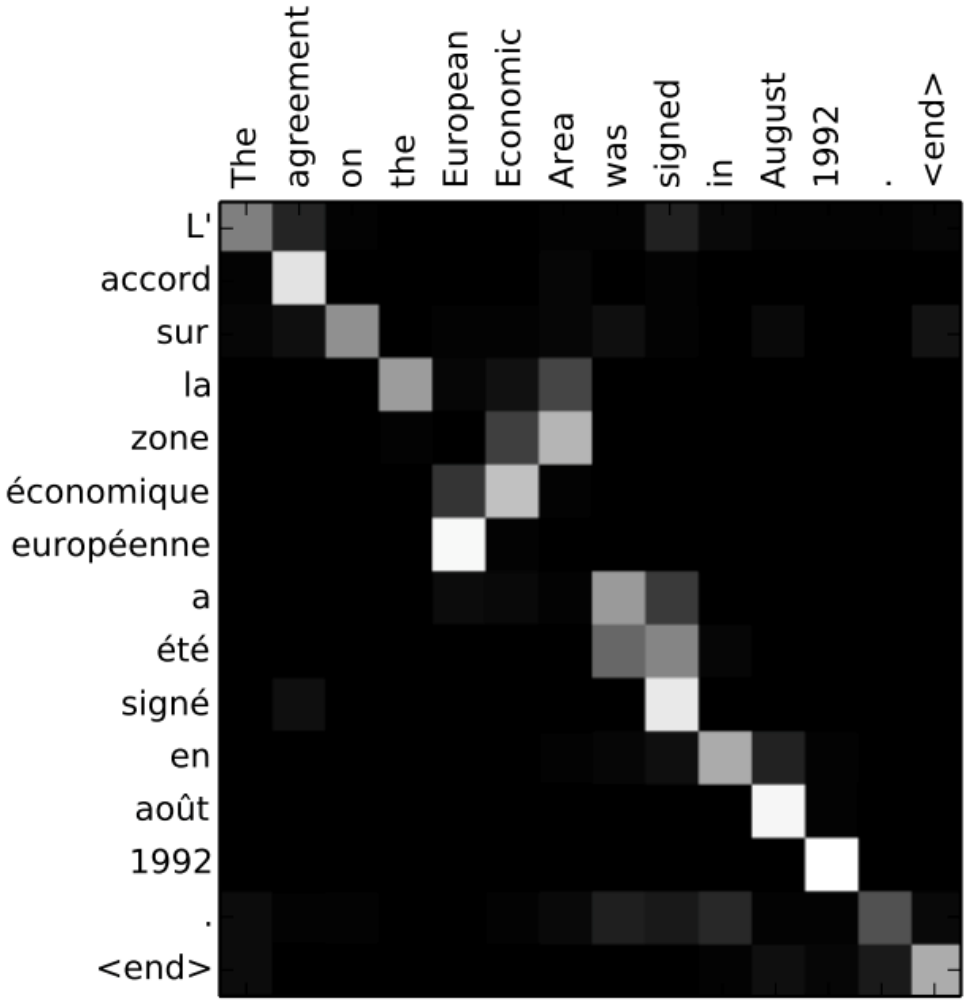
http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Translation



http://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Example


The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>

Attention is all you need

Attention Is All You Need
https://arxiv.org › cs ▾ 翻译此页
作者：A Vaswani - 2017 - 被引用次数：668 - 相关文章
2017年6月12日 - The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network ...

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Visual Attention

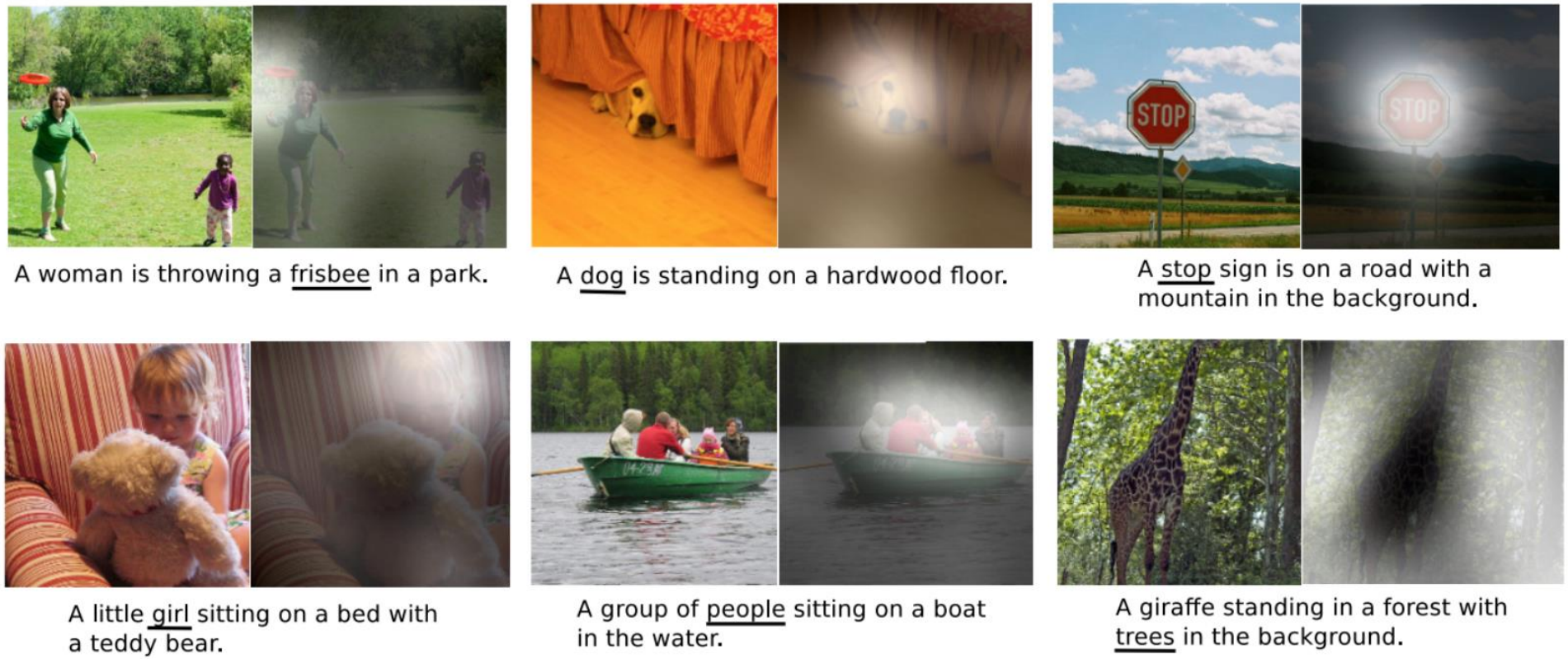
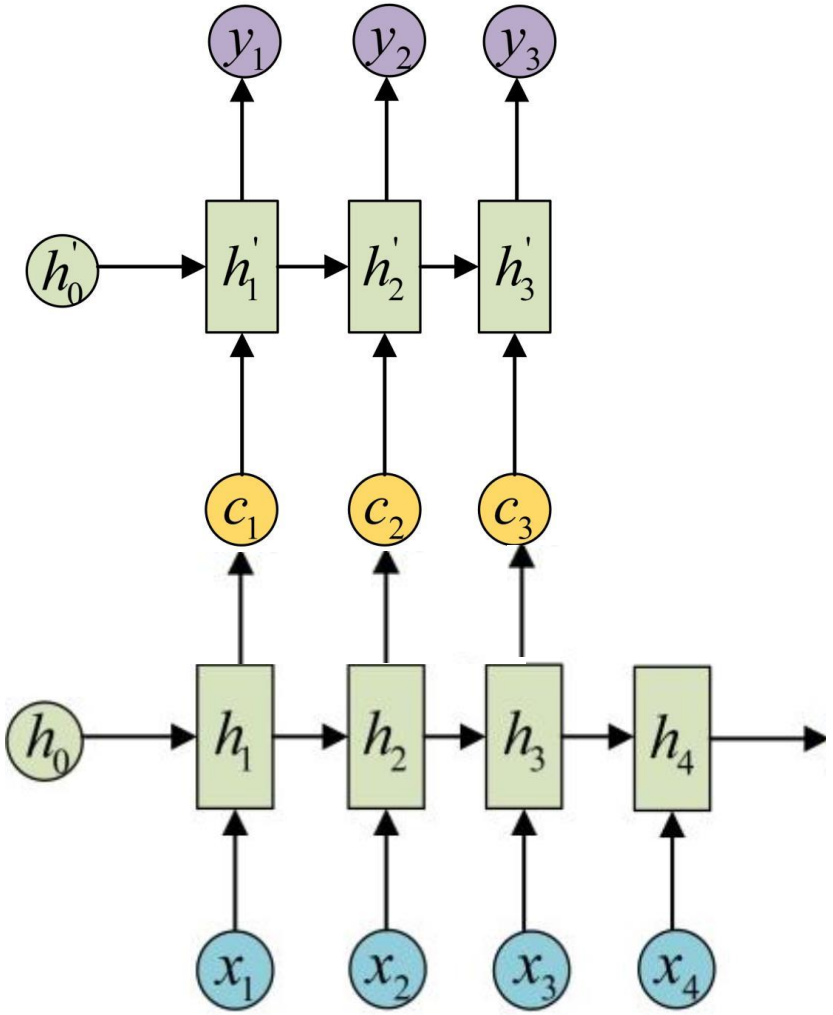
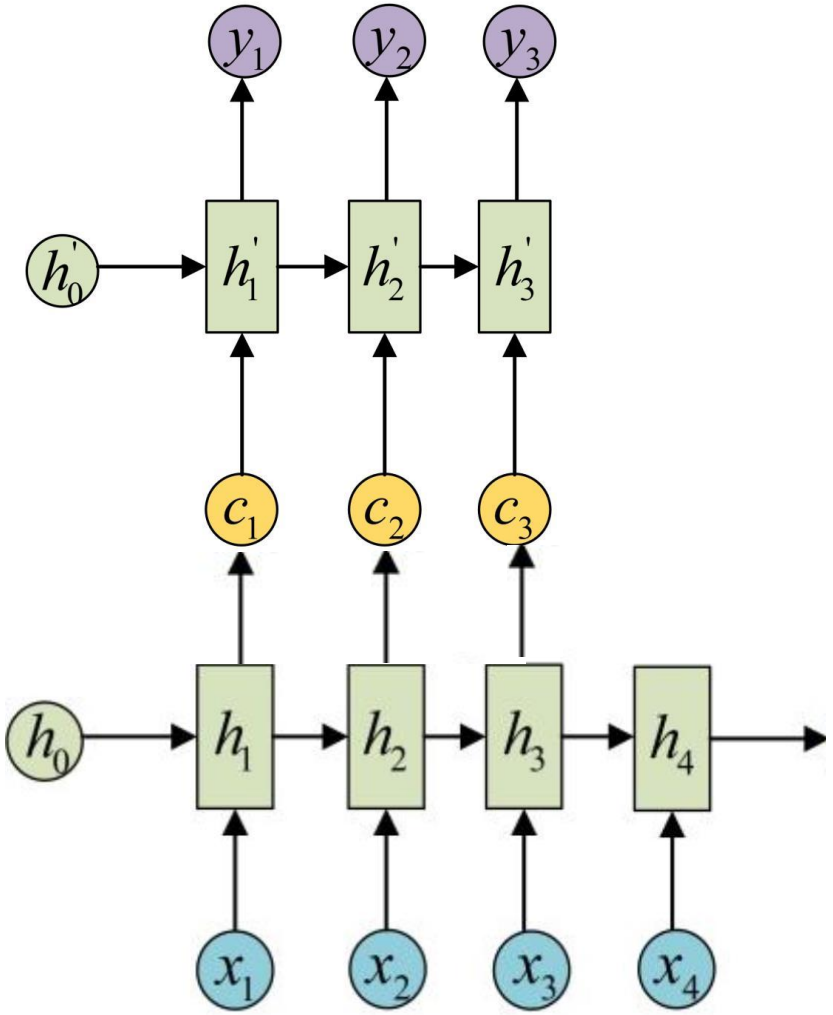


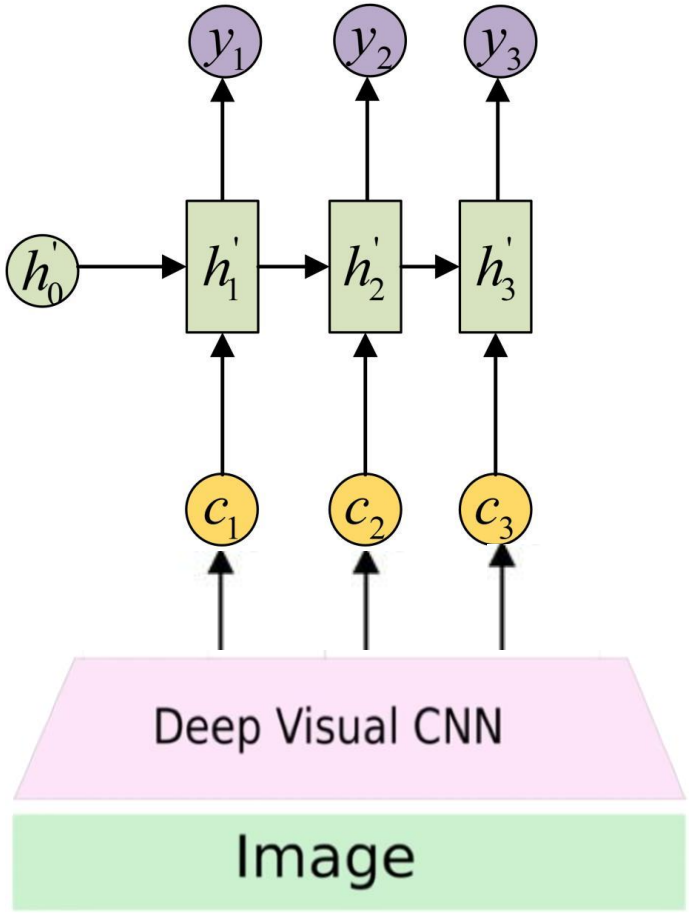
Figure 6: Attending to objects in an image during caption generation. The white regions indicate where the attention mechanism focused on during the generation of the underlined word. From Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

Visual Attention



Visual Attention





Visual Attention

Girl throwing a frisbee

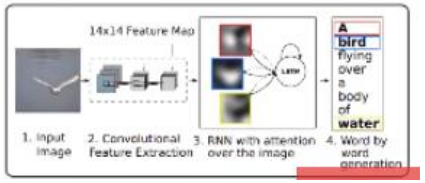
Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Kelvin Xu KELVIN.XU@UMONTREAL.CA
Jimmy Lei Ba JIMMY@PSI.UTORONTO.CA
Ryan Kiros RKIROS@CS.TORONTO.EDU
Kyunghyun Cho KYUNGHYUN.CHO@UMONTREAL.CA
Aaron Courville AARON.COURVILLE@UMONTREAL.CA
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
Yoshua Bengio FIND-ME@THE.WEB

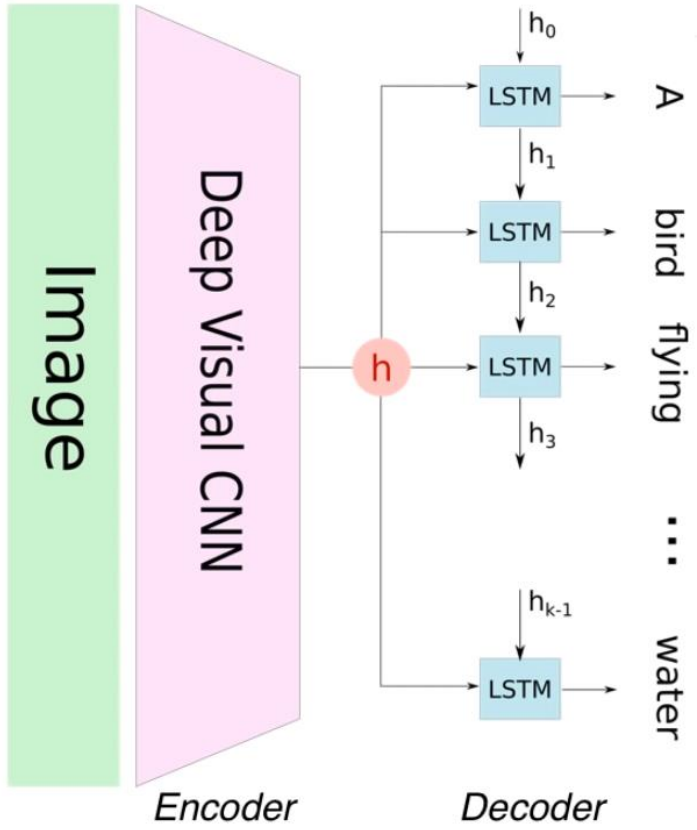
Abstract

Inspired by recent work in machine translation and object detection, we introduce an attention based model that automatically learns to describe the content of images. We describe how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. We also show through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the cor-

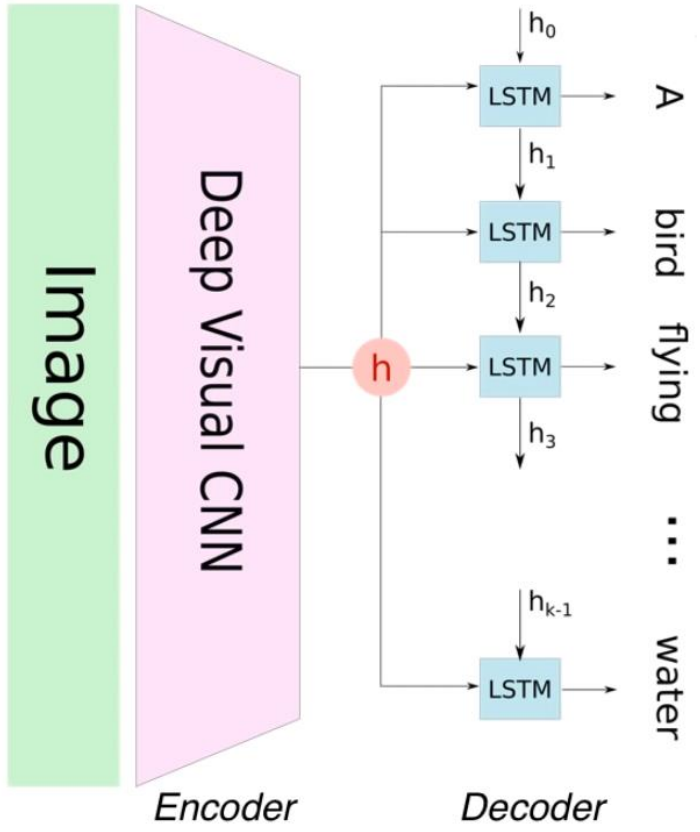
Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4

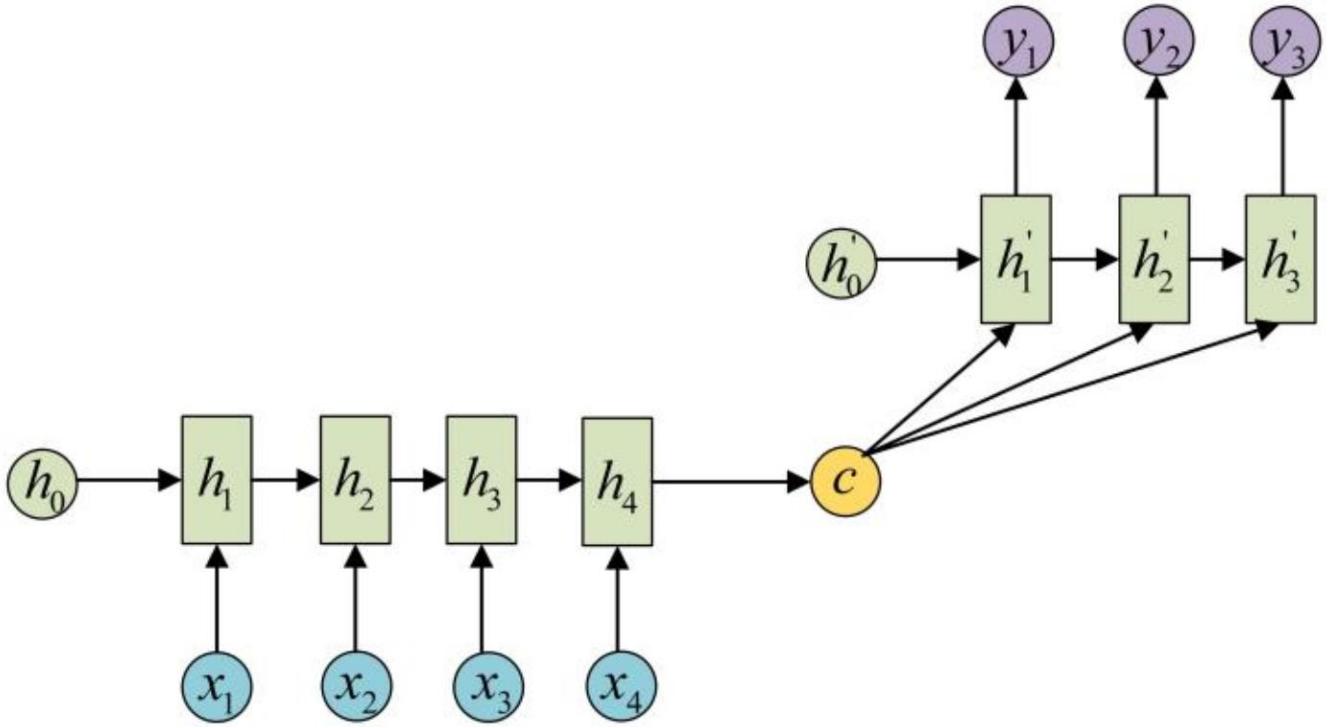


Visual Attention

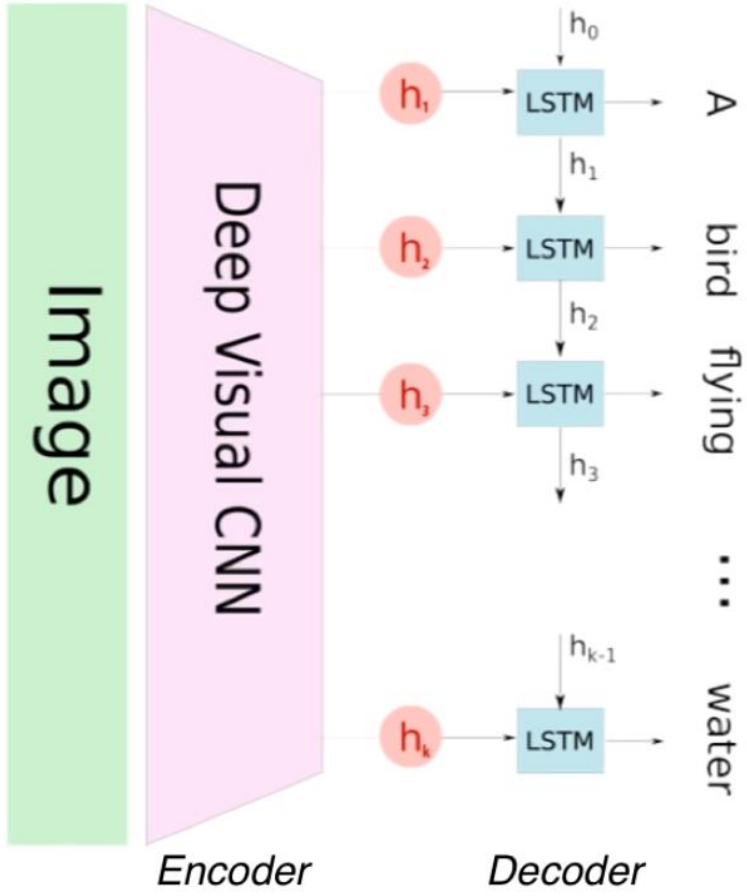


Visual Attention



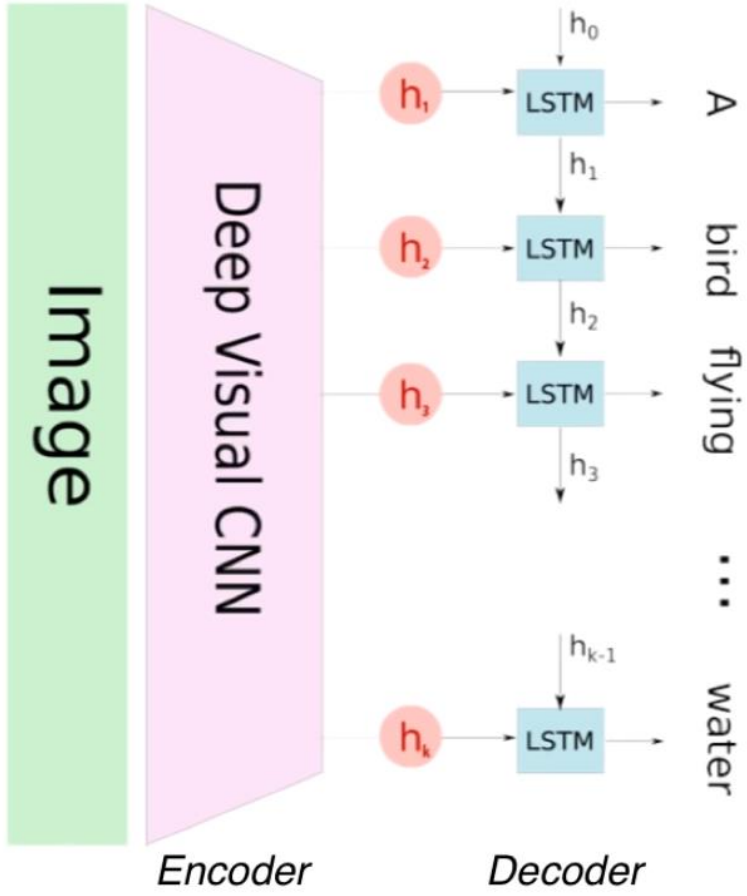


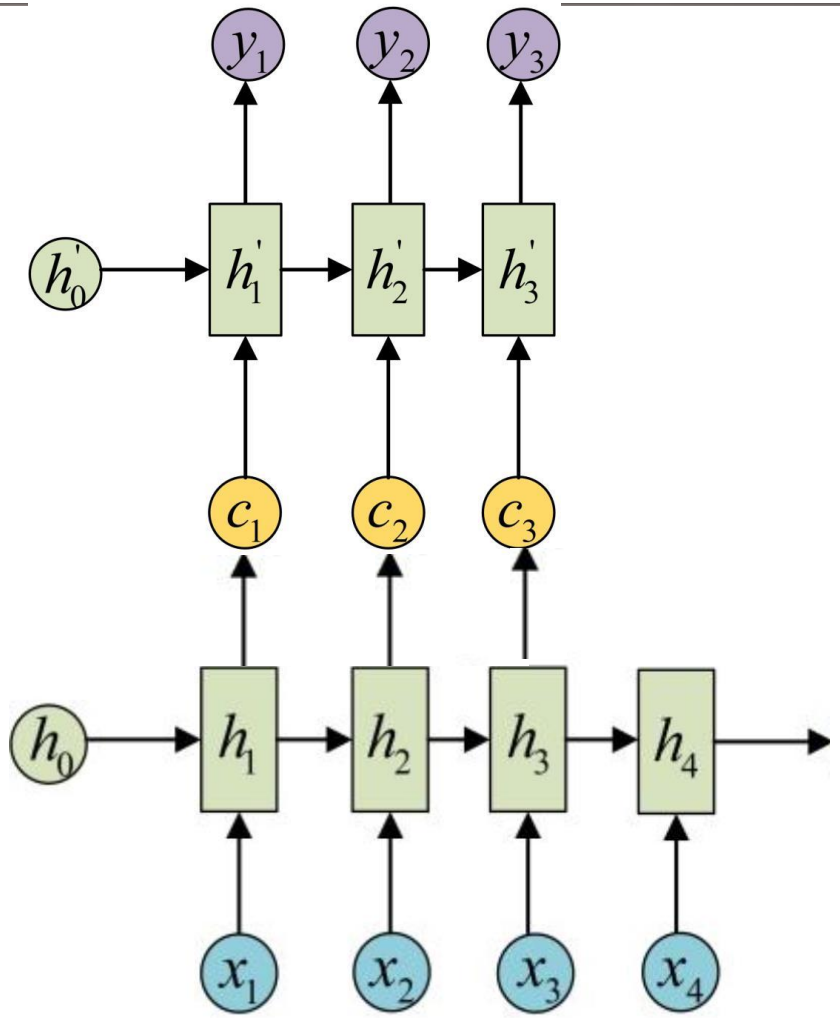
Visual Attention



A Girl throws a Frisbee in the park.

Visual Attention





Paper

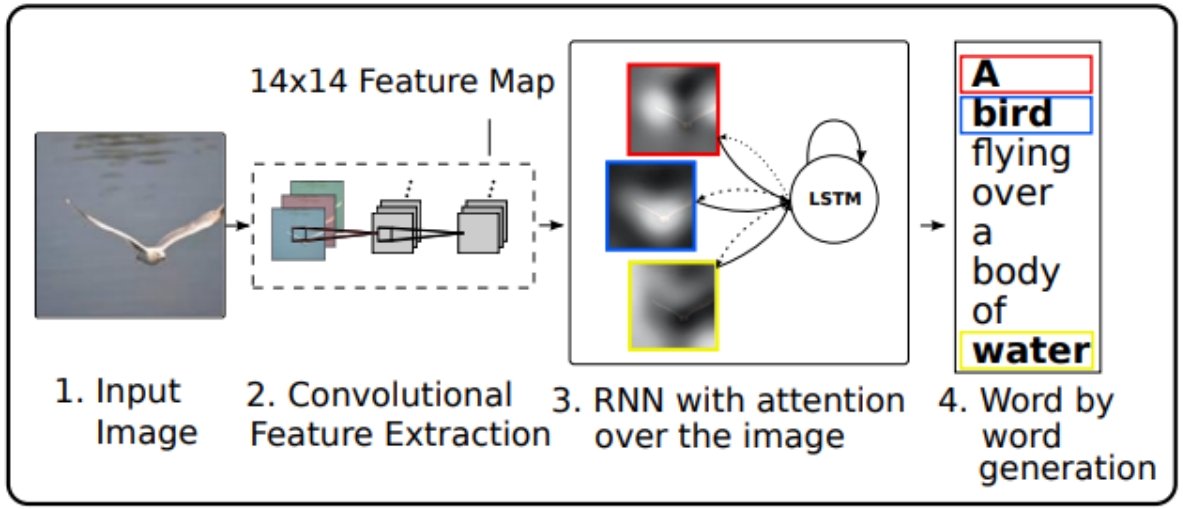
Show, Attend and Tell: Neural Image Caption Generation with Visual ...
https://arxiv.org › cs ▾ 翻译此页
作者：K Xu - 2015 - 被引用次数：2265 - 相关文章
2015年2月10日 - ... Attend and Tell: Neural Image Caption Generation with Visual Attention ... We also show through visualization how the model is able to ...

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

---

Kelvin Xu KELVIN.XU@UMONTREAL.CA
Jimmy Lei Ba JIMMY@PSI.UTORONTO.CA
Ryan Kiros RKIROS@CS.TORONTO.EDU
Kyunghyun Cho KYUNGHYUN.CHO@UMONTREAL.CA
Aaron Courville AARON.COURVILLE@UMONTREAL.CA
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
Yoshua Bengio FIND-ME@THE.WEB

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4



Medical Image

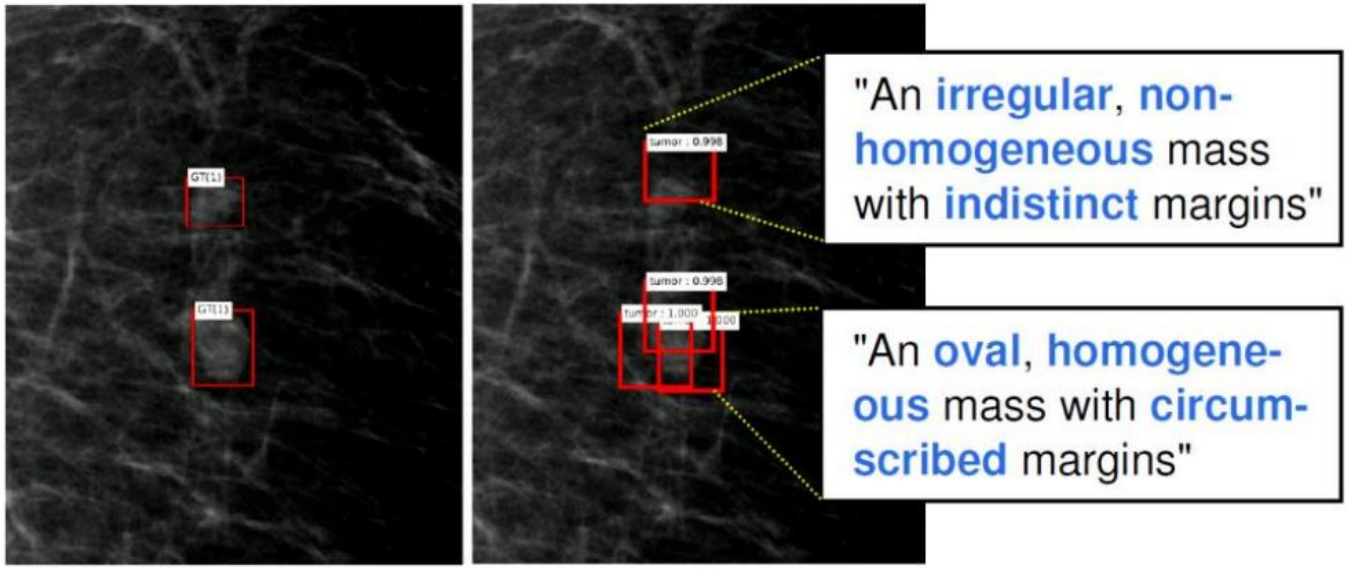
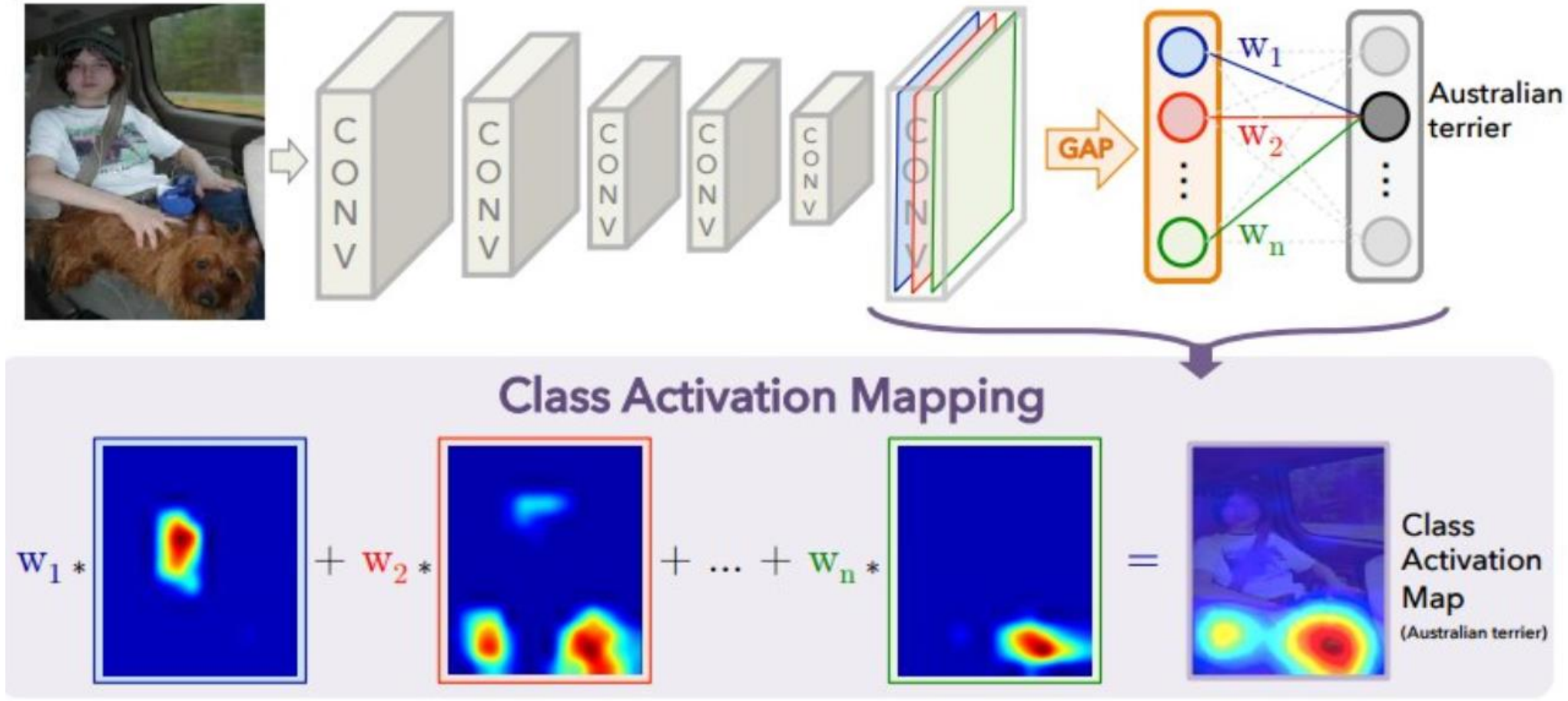
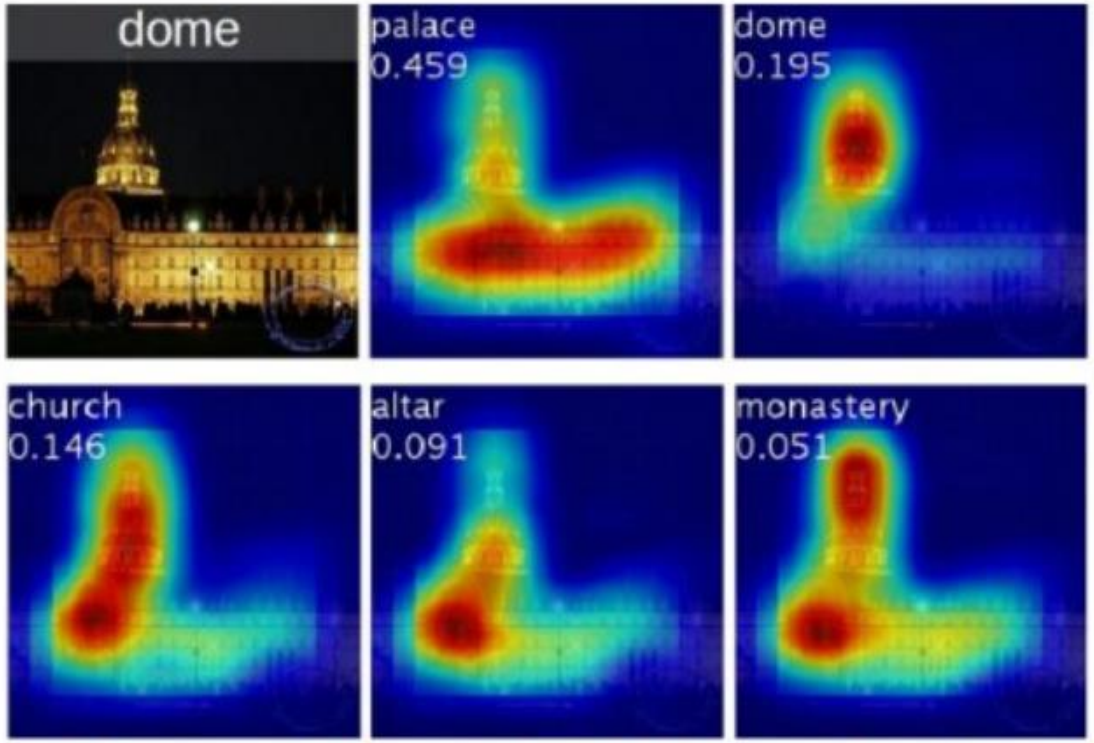


Fig. 1. An example of the output of the proposed multi-task-loss CNN based system. (a) cropped-out mammogram with 2 marked rectangular ground truth areas containing lesions, (b) corresponding top-4 automatically detected bounding box (BB) candidates, (c) automatically generated textual description of lesions in the BB's. The estimated semantic values (in blue) are embedded into predefined sentence templates. The three lower BB's have the same estimated description.

https://dlpm2016.fbk.eu/docs/kisilev_learning.pdf

Class Activation Mapping

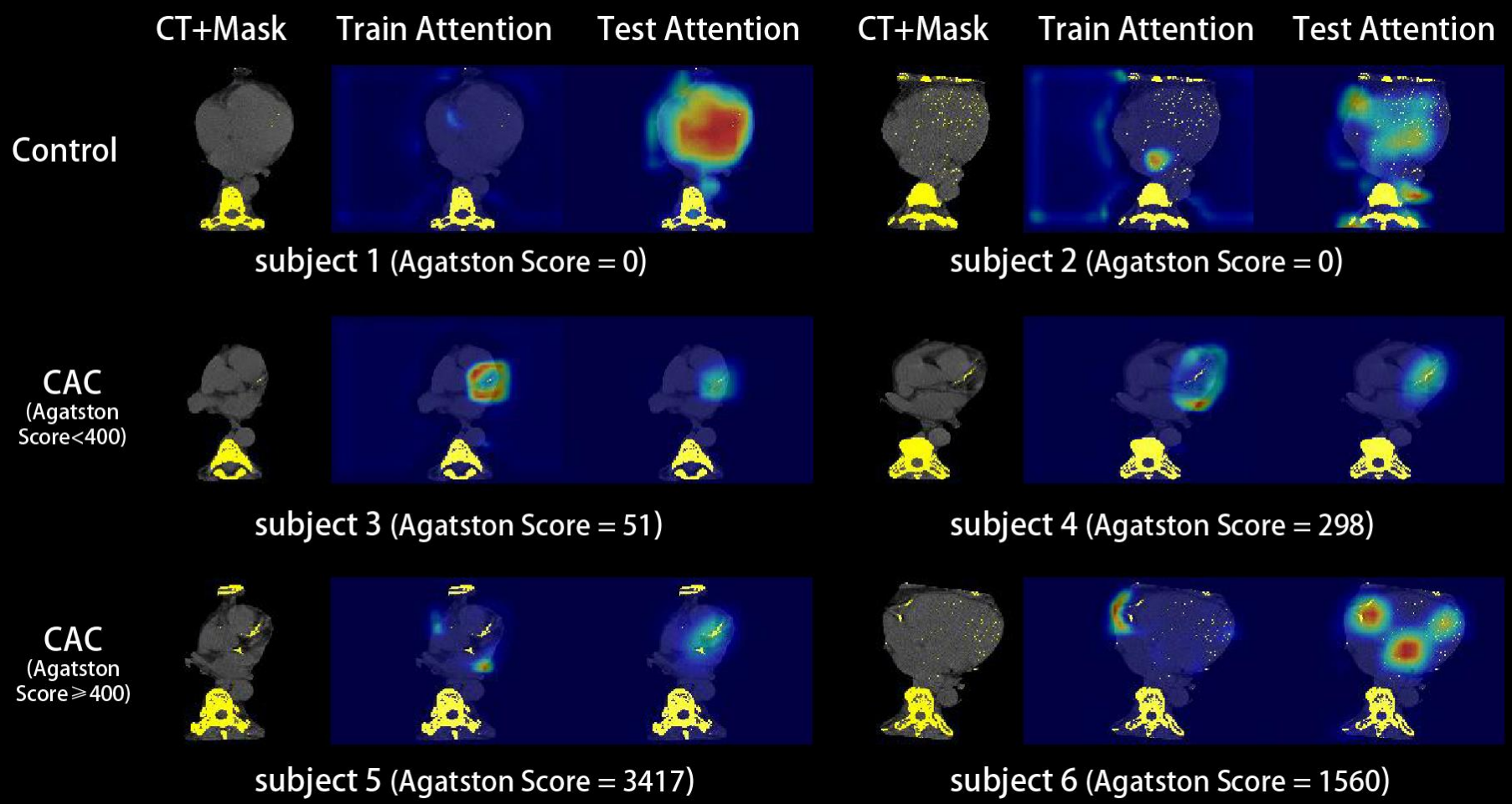




Class activation maps of top 5 predictions

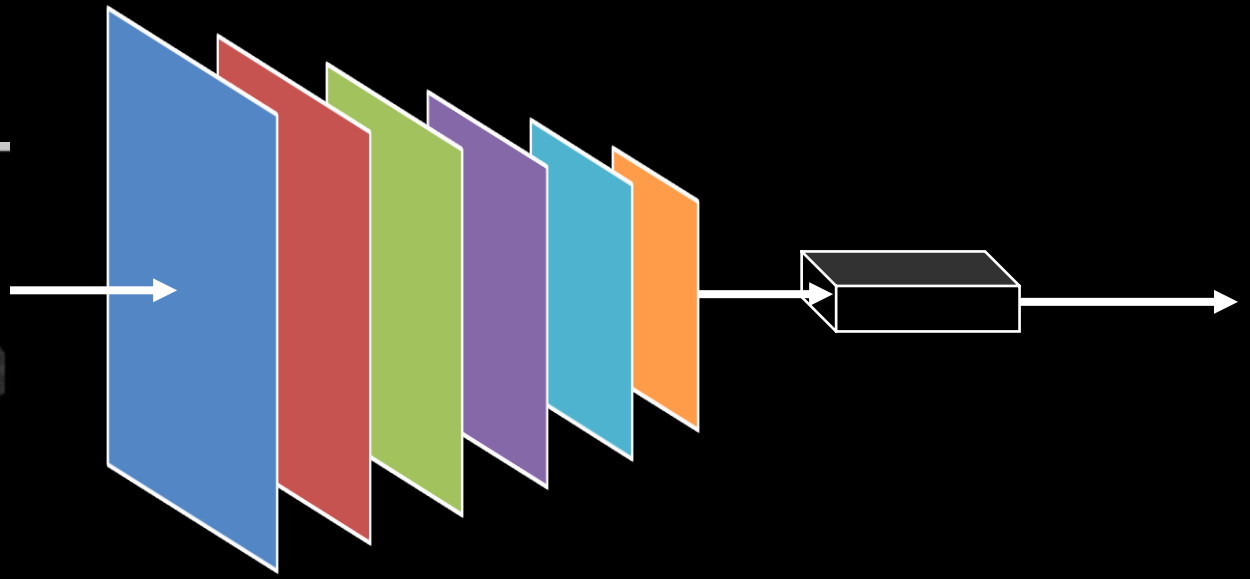
Class activation maps for one object class

Medical Image



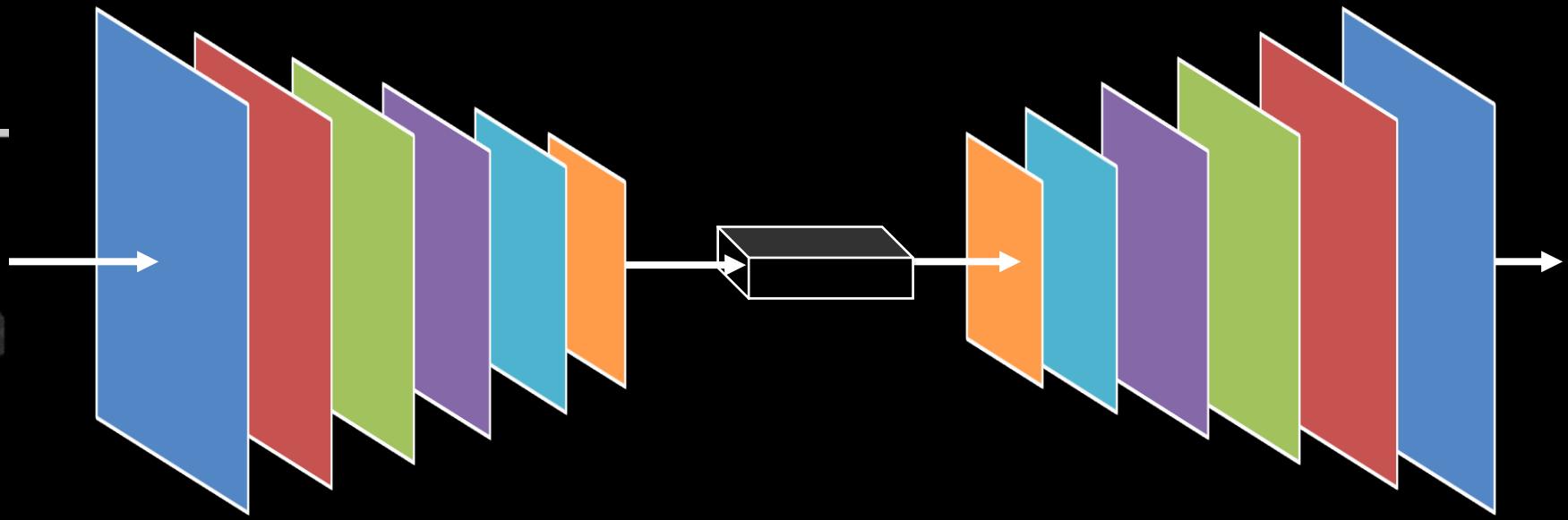
Limitation

Input

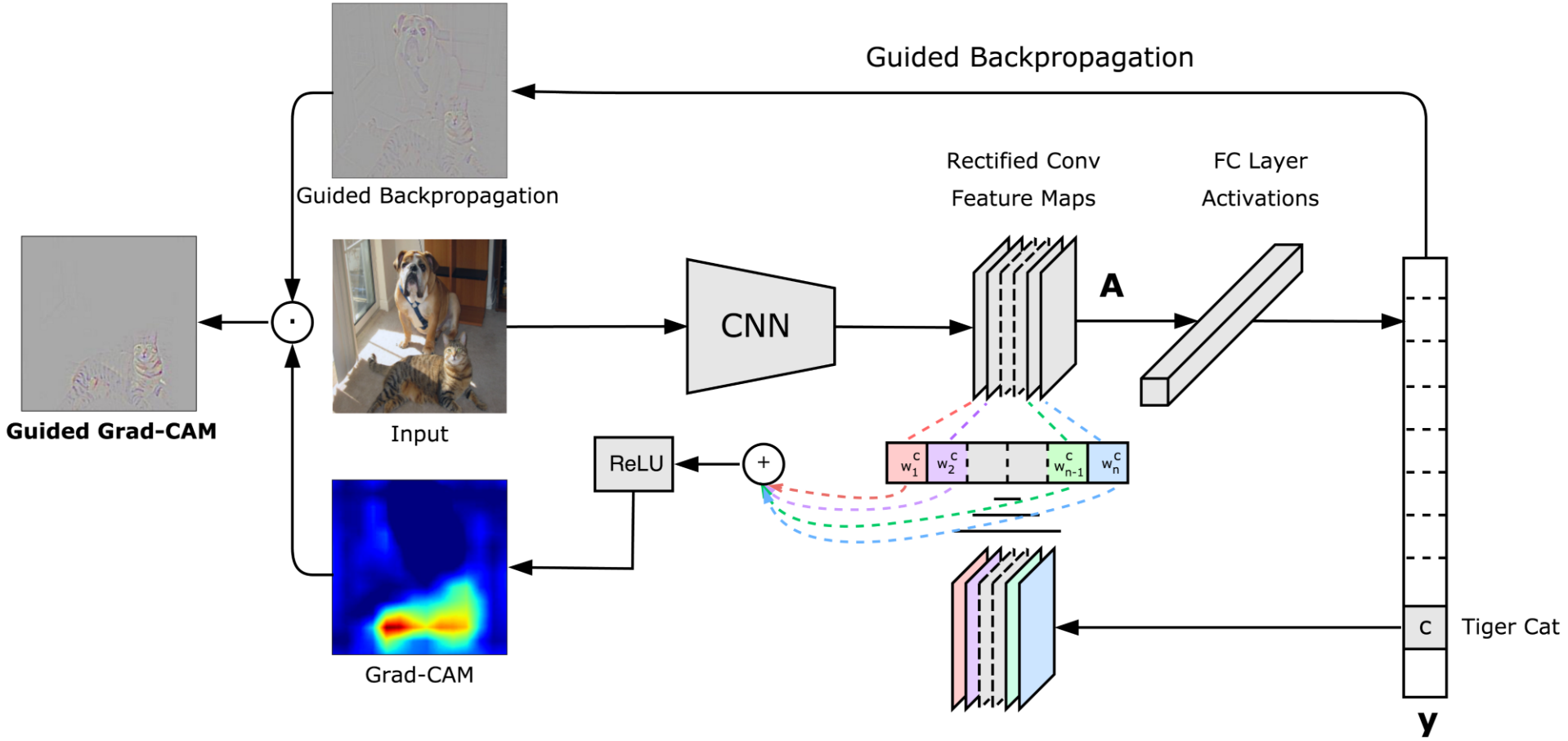


Limitation

Input

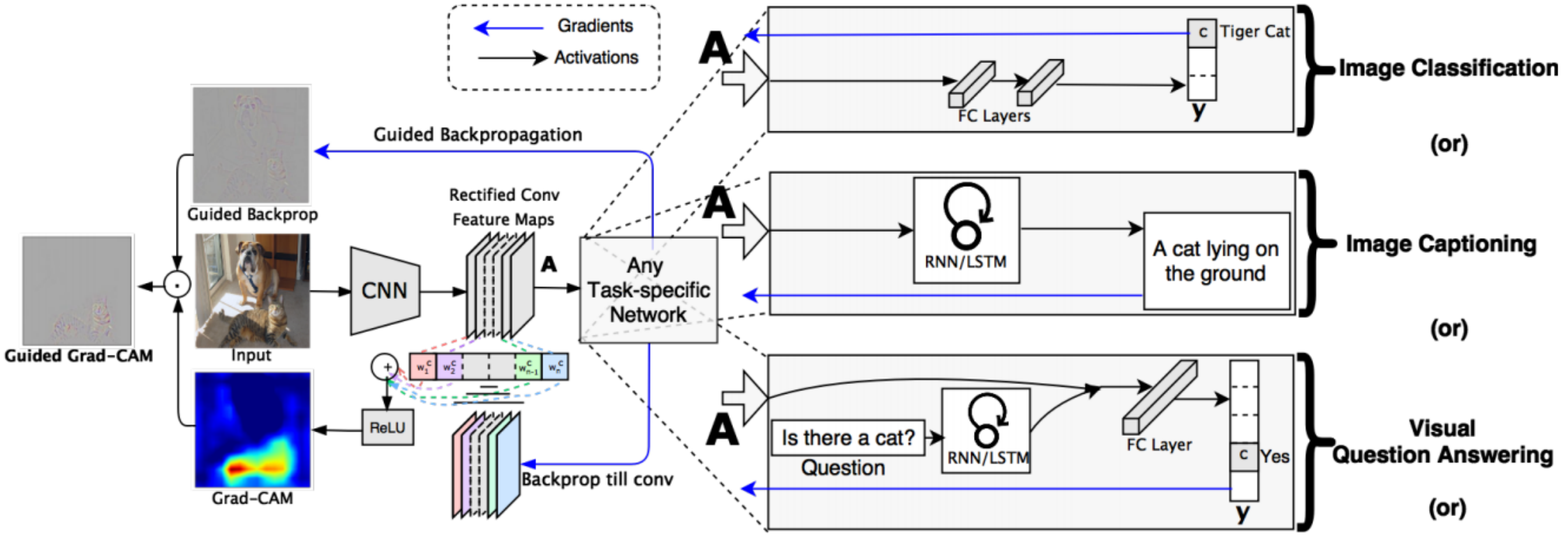


Grad CAM



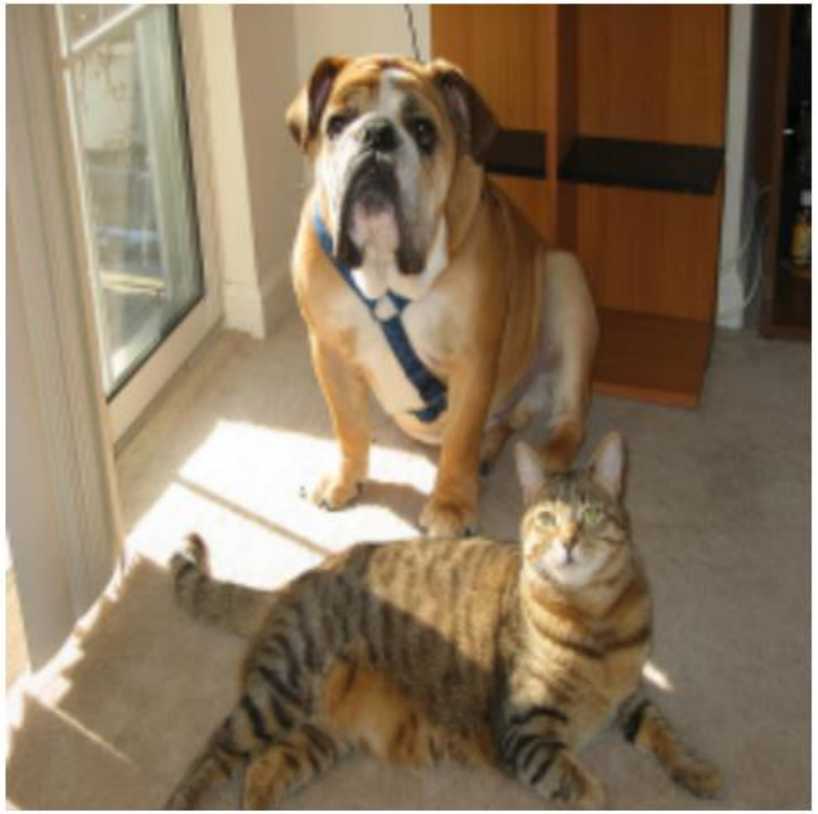
https://arxiv.org/pdf/1610.02391.pdf

GradCAM



https://arxiv.org/pdf/1610.02391.pdf

GradCAM



https://arxiv.org/pdf/1610.02391.pdf

Attention Gate

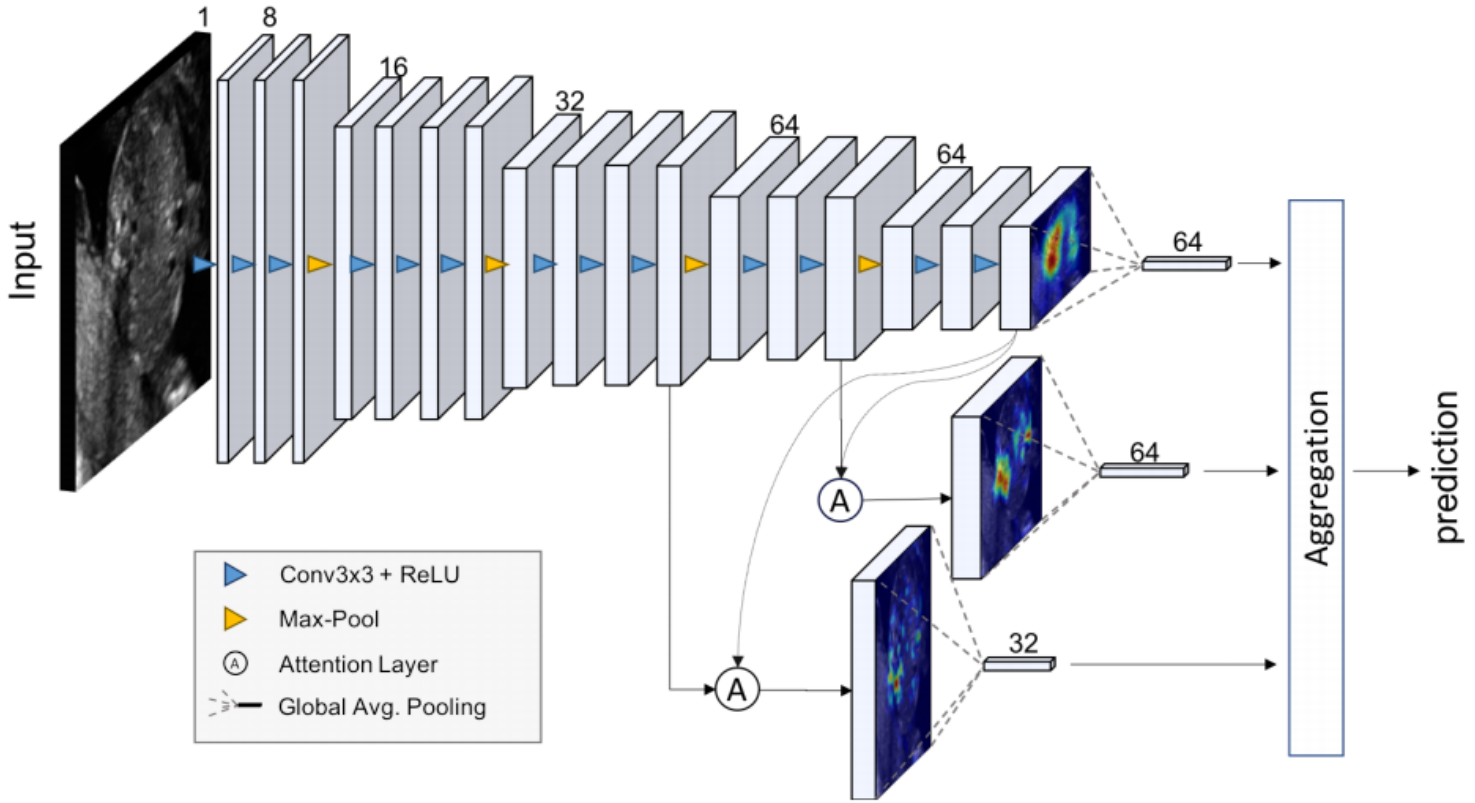


Figure 1: The schematics of the proposed network, termed *Attention-Gated Sononet*. The proposed attention units are incorporated in layer 11 and layer 14.

https://arxiv.org/pdf/1804.05338.pdf

Attention Gate

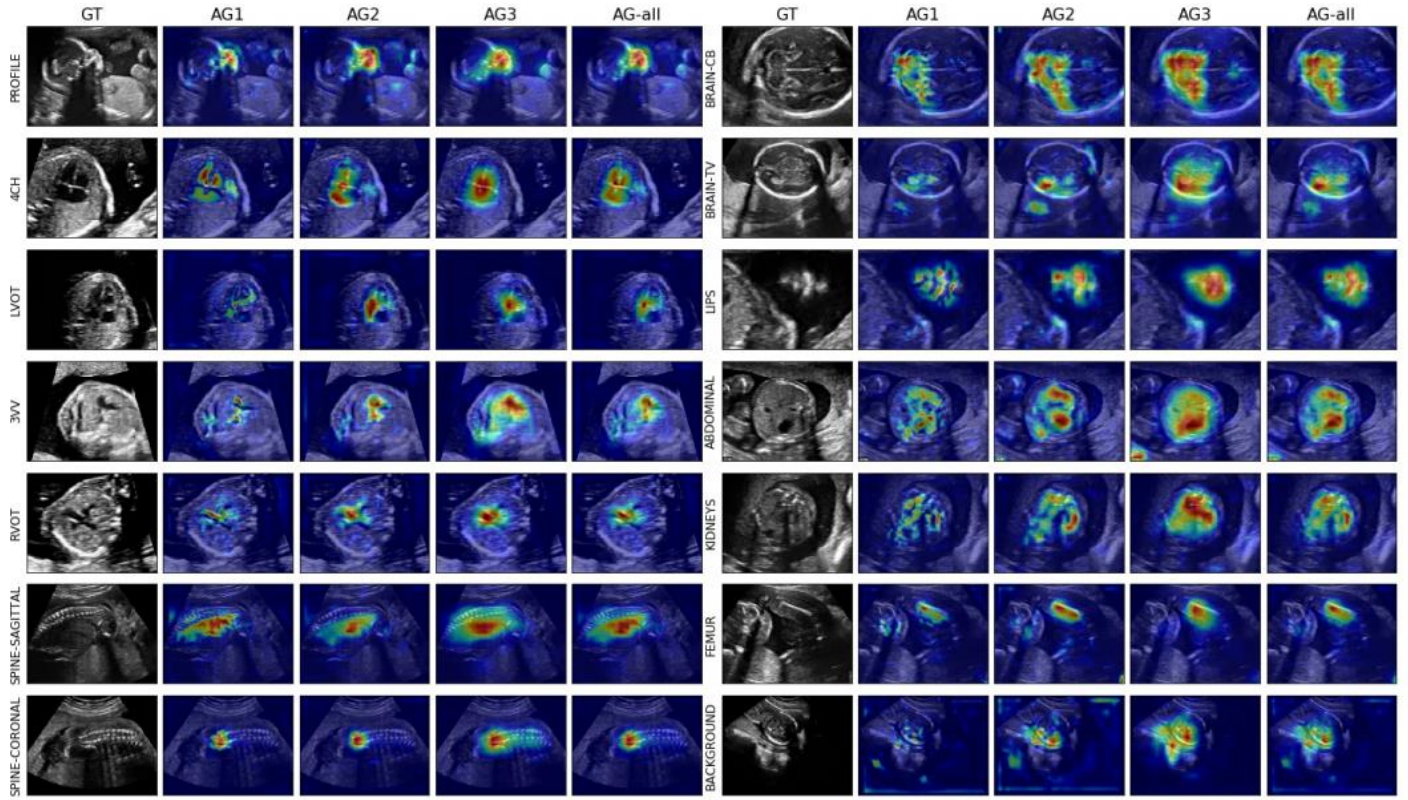
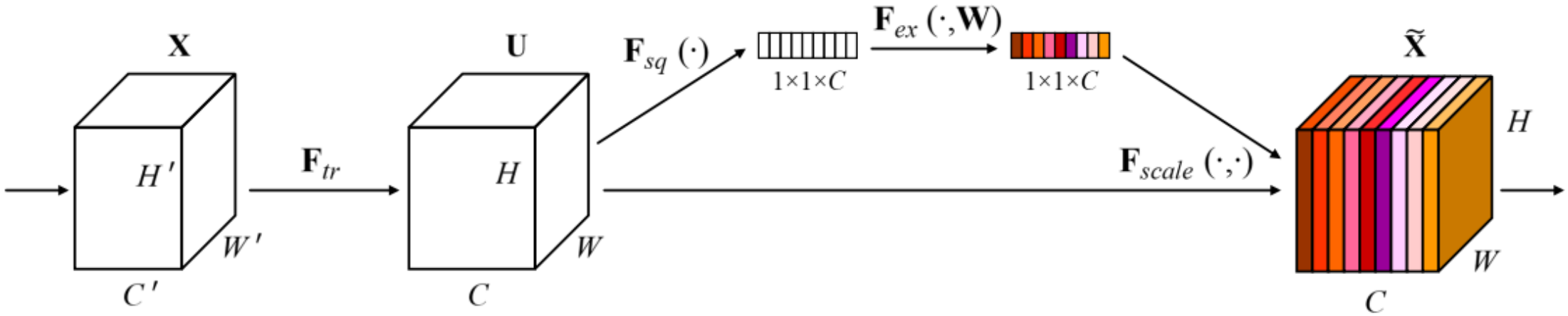


Figure 3: Examples of obtained attention map from AG-sononet. AG1 and AG2 are from layer 11 and 14 respectively. AG3 is obtained using CAM [33]. AG-all is obtained by normalising the maximum attented value across all AG's and taking mean over them.

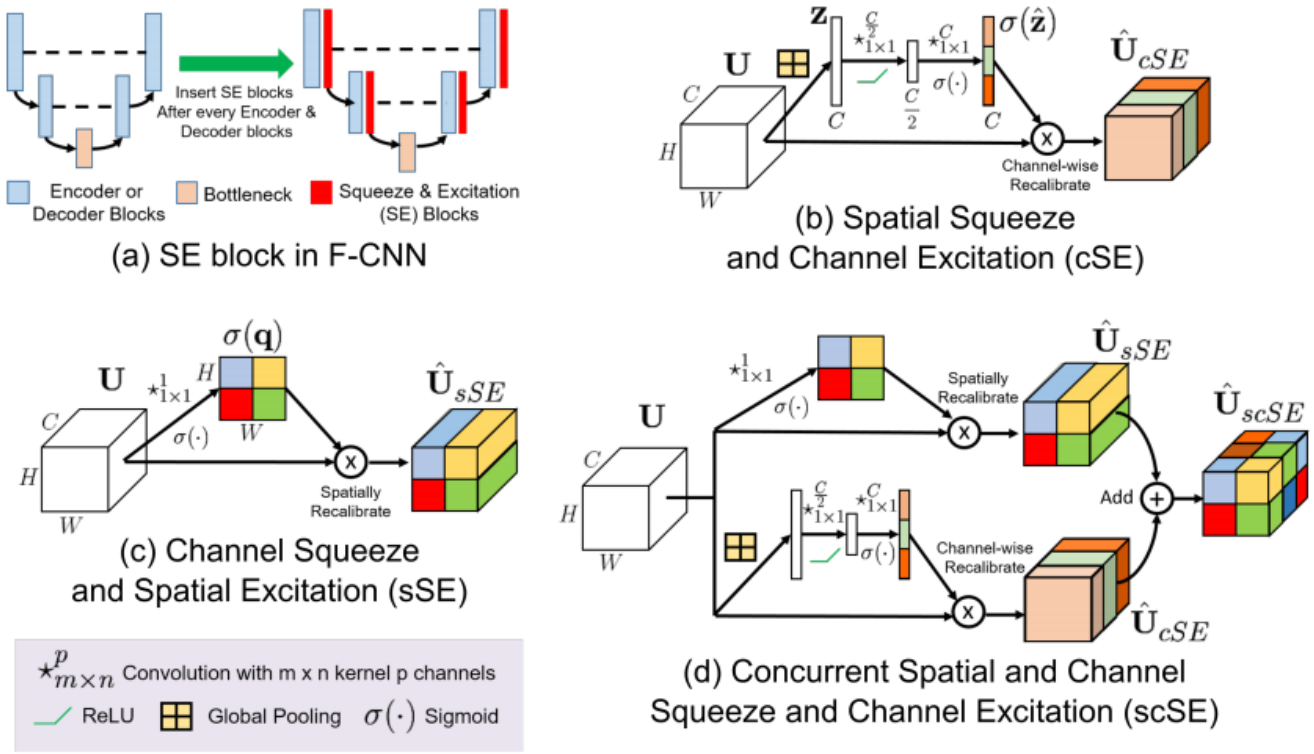
Squeeze-and-Excitation Networks



2017 ILSVR

https://arxiv.org/abs/1709.01507

Medical Imaging



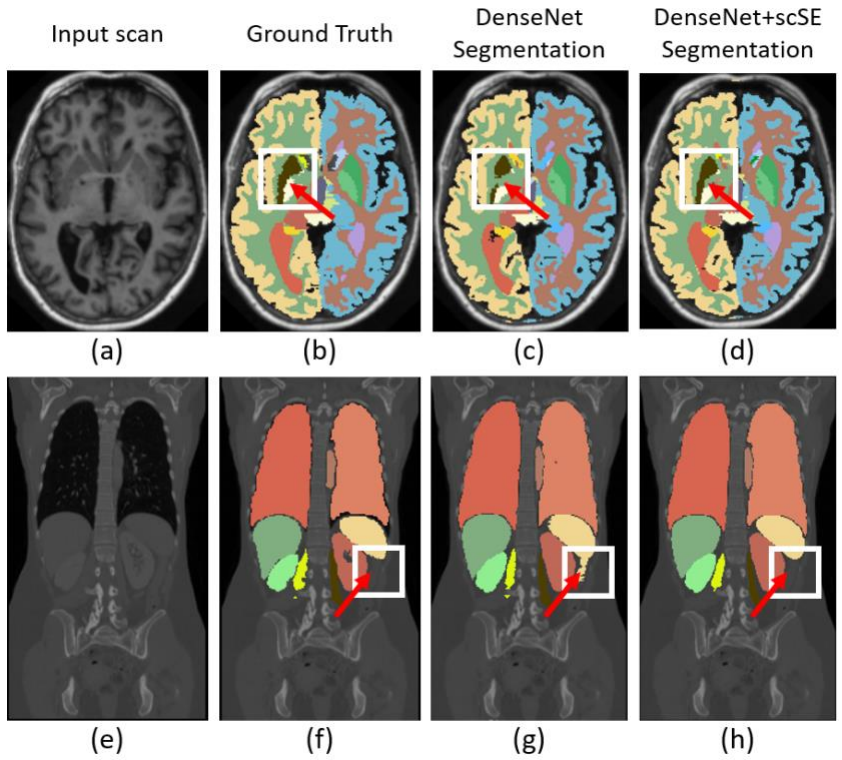


Fig. 1: Illustration of network architecture with squeeze & excitation (SE) blocks. (a) The proposed integration of SE blocks within F-CNN. (b-d) The architectural design of cSE, sSE and scSE blocks, respectively, for recalibrating feature map **U**.

Respond-CAM: Analyzing Deep Models for 3D Imaging Data by Visualizations

Guannan Zhao[1], Bo Zhou[2], Kaiwen Wang[2], Rui Jiang[1], and Min Xu[2(✉)]

[1] Department of Automation, Tsinghua University, Beijing, China
mxu1@cs.cmu.edu
[2] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. The convolutional neural network (CNN) has become a powerful tool for various biomedical image analysis tasks, but there is a lack of visual explanation for the machinery of CNNs. In this paper, we present a novel algorithm, Respond-weighted Class Activation Mapping (Respond-CAM), for making CNN-based models interpretable by visualizing input regions that are important for predictions, especially for biomedical 3D imaging data inputs. Our method uses the gradients of any target concept (e.g. the score of target class) that flow into a convolutional layer. The weighted feature maps are combined to produce a heatmap that highlights the important regions in the image for predicting the target concept. We prove a preferable sum-to-score property of the Respond-CAM and verify its significant improvement on 3D images from the current state-of-the-art approach. Our tests on Cellular Electron Cryo-Tomography 3D images show that Respond-CAM achieves superior performance on visualizing the CNNs with 3D biomedical image inputs, and is able to get reasonably good results on visualizing the CNNs with natural image inputs. The Respond-CAM is an efficient and reliable approach for visualizing the CNN machinery, and is applicable to a wide variety of CNN model families and image analysis tasks. Our code is available at: https://github.com/xulabs/projects/tree/master/respond_cam.

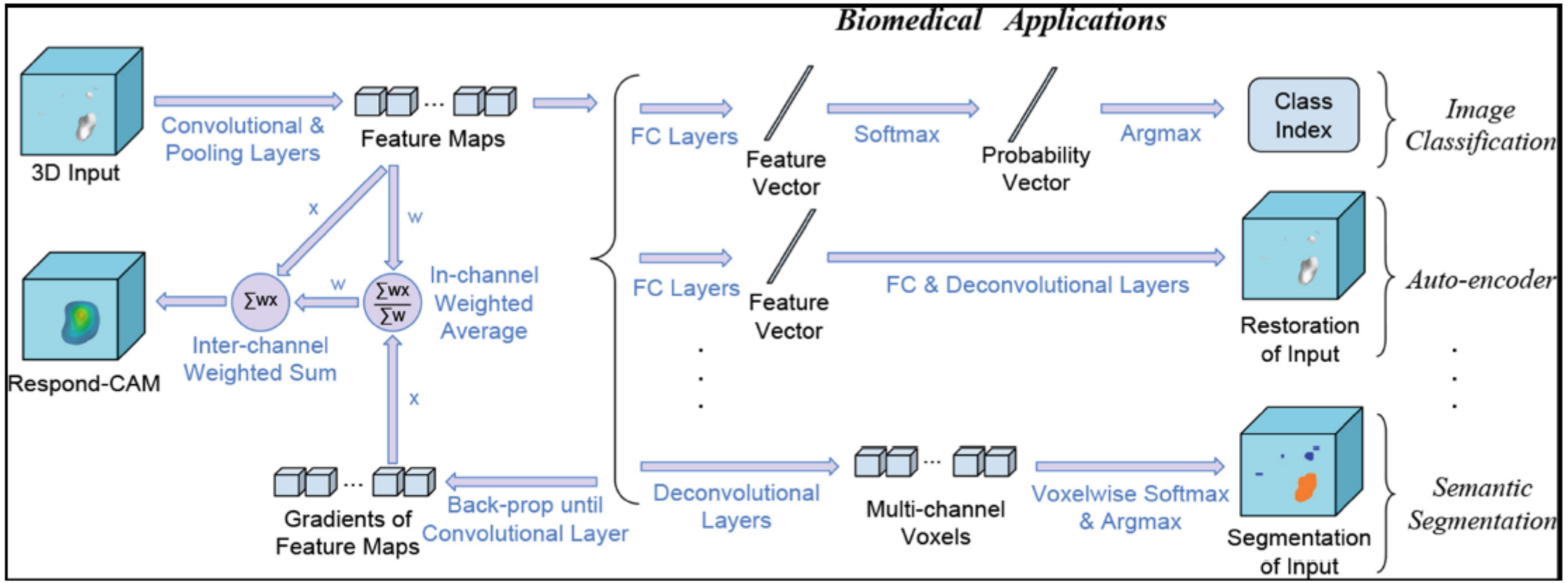


Fig. 1. Overview: for a trained CNN, we choose a differentiable scalar output of interest y (e.g. an entry of the feature vector) and calculate the gradient of y to the feature maps. Then we combine the feature maps and their gradients to get the Respond-CAM representing which areas in the input contribute the most to the output y.

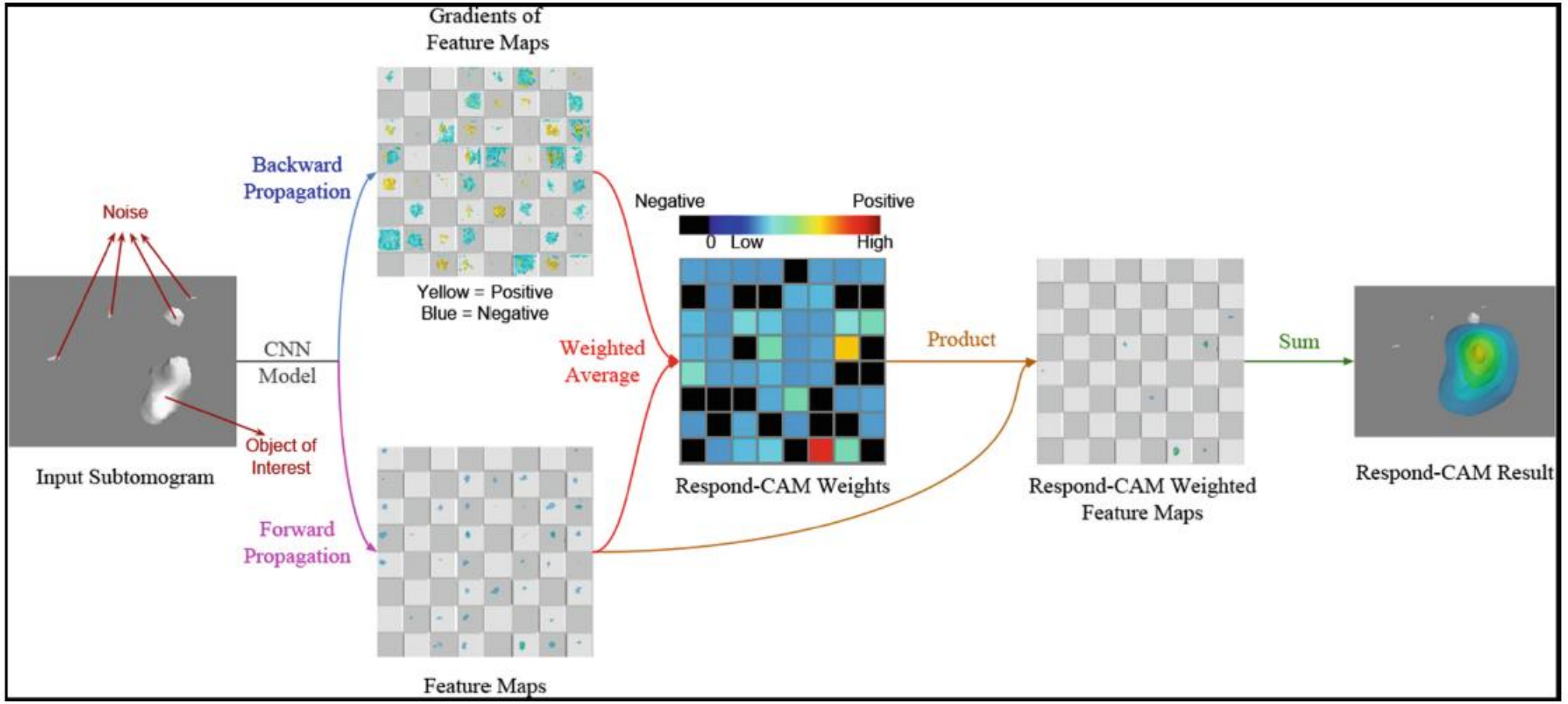


Fig. 2. Respond-CAM walk-through with an example. By applying the weighted average with each feature maps to their corresponding gradients, we get the Respond-CAM weights for each feature map. We calculate the Respond-CAM using Eq. 2 where the map with the largest weight (marked with red) contributes the most. As a result, the object-of-interest area gets highlighted.

Table 1. Comparison between Grad-CAM and Respond-CAM on L1 error and Kendall's Tau (KT) on different CNNs and datasets.

Model	CNN-1	CNN-2	CNN-1	CNN-2
SNR	$+\infty$	$+\infty$	0.1	0.1
L1, Grad	15.344 ± 7.789	8.117 ± 3.854	21.142 ± 8.511	14.020 ± 5.232
L1, Respond	$\mathbf{0.601} \pm 0.187$	$\mathbf{0.770} \pm 0.366$	$\mathbf{1.006} \pm 0.721$	$\mathbf{0.542} \pm 0.194$
KT, Grad	0.273 ± 0.350	0.822 ± 0.097	-0.116 ± 0.341	0.458 ± 0.260
KT, Respond	$\mathbf{0.981} \pm 0.014$	$\mathbf{0.976} \pm 0.018$	$\mathbf{0.975} \pm 0.024$	$\mathbf{0.984} \pm 0.013$

Subject2Vec: Generative-Discriminative Approach from a Set of Image Patches to a Vector

Sumedha Singla[1], Mingming Gong[2], Siamak Ravanbakhsh[3], Frank Sciurba[4], Barnabas Poczos[5], and Kayhan N. Batmanghelich[1,2,5(✉)]

[1] Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA
kayhan@pitt.edu
[2] Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA, USA
[3] Computer Science Department, University of British Columbia, Vancouver, Canada
[4] University of Pittsburgh School of Medicine, University of Pittsburgh, Pittsburgh, PA, USA
[5] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We propose an attention-based method that aggregates local image features to a subject-level representation for predicting disease severity. In contrast to classical deep learning that requires a fixed dimensional input, our method operates on a *set* of image patches; hence it can accommodate variable length input image without image resizing. The model learns a clinically interpretable subject-level representation that is reflective of the disease severity. Our model consists of three mutually dependent modules which regulate each other: (1) a *discriminative* network that learns a fixed-length representation from local features and maps them to disease severity; (2) an *attention* mechanism that provides interpretability by focusing on the areas of the anatomy that contribute the most to the prediction task; and (3) a *generative* network that encourages the diversity of the local latent features. The generative term ensures that the attention weights are non-degenerate while maintaining the relevance of the local regions to the disease severity. We train our model end-to-end in the context of a large-scale lung CT study of Chronic Obstructive Pulmonary Disease (COPD). Our model gives state-of-the art performance in predicting clinical measures of severity for COPD. The distribution of the attention provides the regional relevance of lung tissue to the clinical measurements.

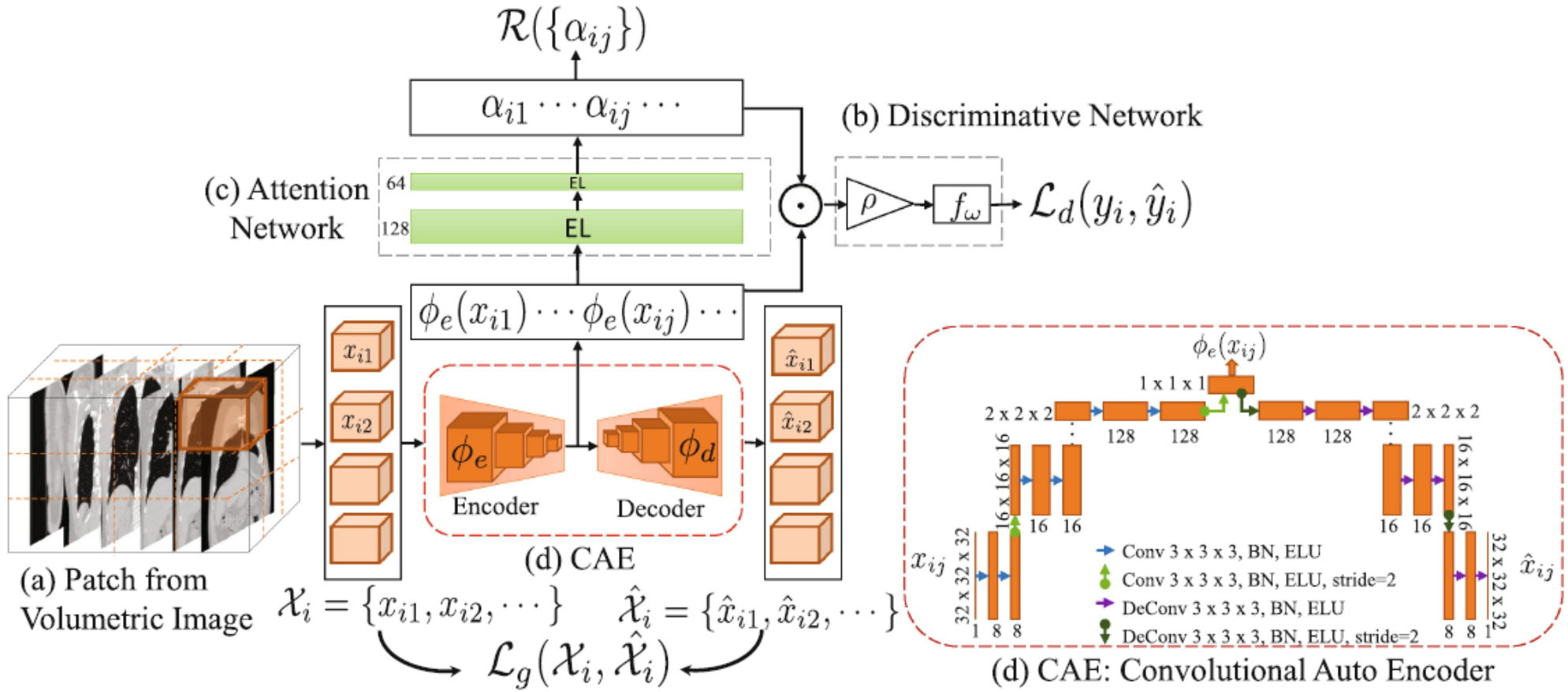


Fig. 1. (a) A subject is represented as a set of 3d image patches, (b) Discriminative Network: aggregates local features to form a fixed length representation for the subject and predicts the disease severity, (c) Attention Network: focuses attention on critical patches to provide interpretability, (d) Convolutional Auto Encoder (Generative Network): prevents redundancy of latent features.

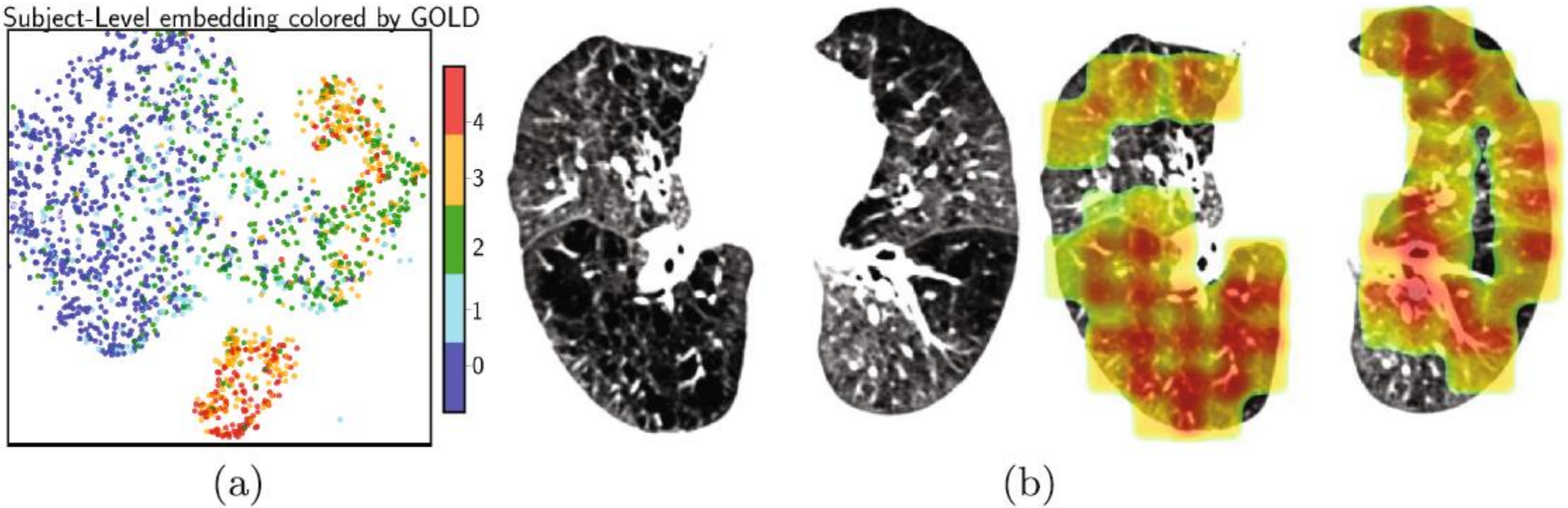
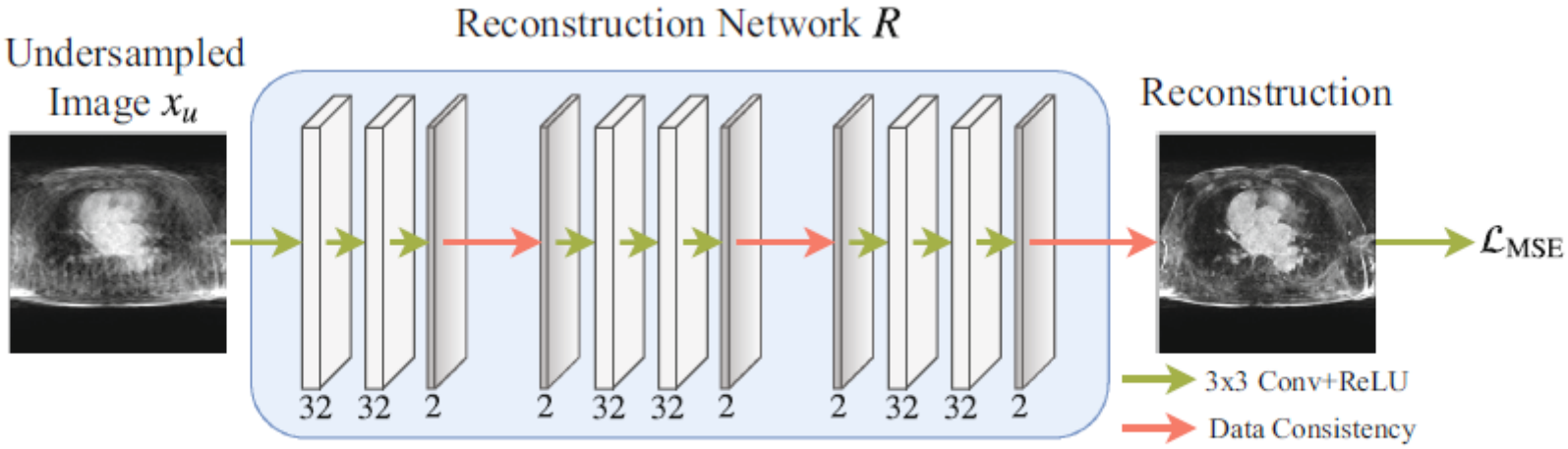


Fig. 3. (a) Embedding the subjects in 2D using tSNE. The dots represents one subject colored by the GOLD score. (b) An axial view of the attention map on a subject. Red color indicate higher relevance to the disease severity.

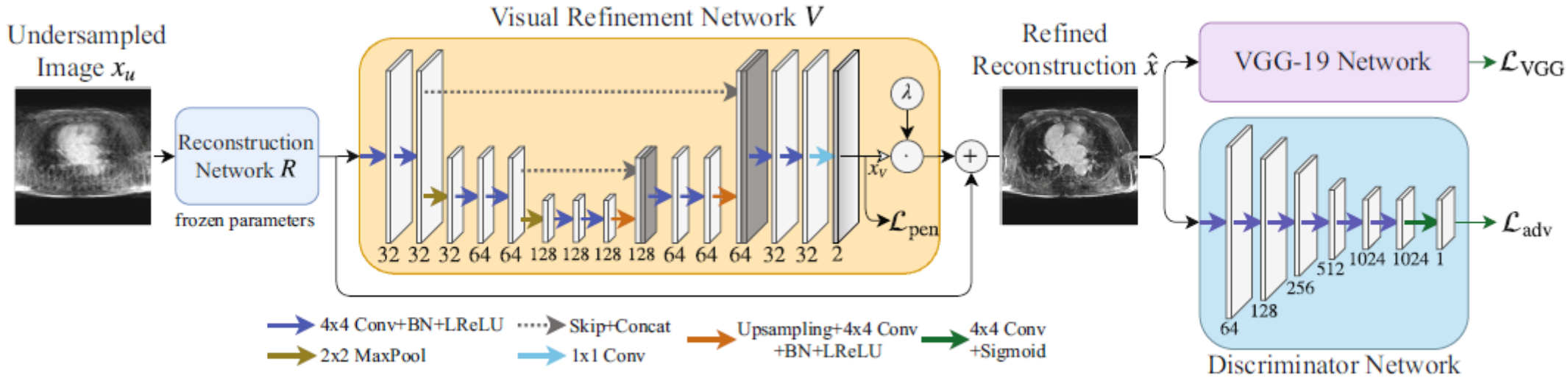
Adversarial and Perceptual Refinement for Compressed Sensing MRI Reconstruction

Maximilian Seitzer[1,2(✉)], Guang Yang[3,4], Jo Schlemper[2], Ozan Oktay[2], Tobias Würfl[1], Vincent Christlein[1], Tom Wong[3,4], Raad Mohiaddin[3,4], David Firmin[3,4], Jennifer Keegan[3,4], Daniel Rueckert[2], and Andreas Maier[1]

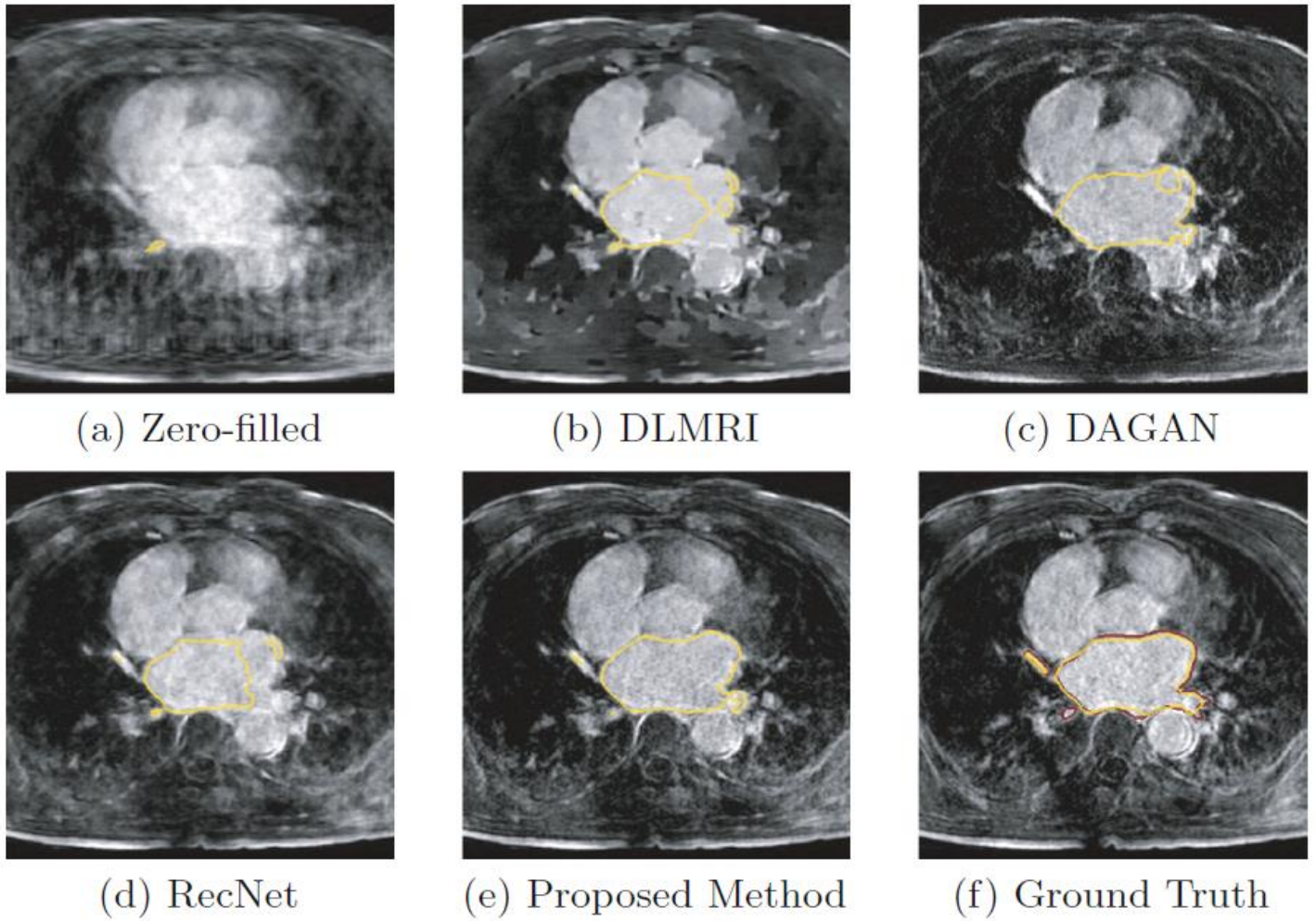
[1] Pattern Recognition Lab, Friedrich-Alexander-Universität, Erlangen, Germany
maximilian.seitzer@fau.de
[2] Biomedical Image Analysis Group, Imperial College, London, UK
[3] National Heart and Lung Institute, Imperial College, London, UK
[4] Cardiovascular Research Centre, Royal Brompton Hospital, London, UK



(a) Stage 1: training of reconstruction network using $\mathcal{L}_{\text{MSE}}(R)$.



(b) Stage 2: training of visual refinement network using $\mathcal{L}_{\text{vis}}(V)$.



(a) Zero-filled (b) DLMRI (c) DAGAN

(d) RecNet (e) Proposed Method (f) Ground Truth

Fig. 2. Qualitative visualization for 8-fold undersampling. Contour of predicted segmentation of left atrium in yellow, contour of ground truth segmentation in red.

Adversarial Sparse-View CBCT Artifact Reduction

Haofu Liao[1(✉)], Zhimin Huo[1], William J. Sehnert[2], Shaohua Kevin Zhou[3], and Jiebo Luo[1]

[1] Department of Computer Science, University of Rochester, Rochester, USA
hliao6@cs.rochester.edu
[2] Carestream Health Inc., Rochester, USA
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

Abstract. We present an effective post-processing method to reduce the artifacts from sparsely reconstructed cone-beam CT (CBCT) images. The proposed method is based on the state-of-the-art, image-to-image generative models with a perceptual loss as regulation. Unlike the traditional CT artifact-reduction approaches, our method is trained in an adversarial fashion that yields more perceptually realistic outputs while preserving the anatomical structures. To address the streak artifacts that are inherently local and appear across various scales, we further propose a novel discriminator architecture based on feature pyramid networks and a differentially modulated focus map to induce the adversarial training. Our experimental results show that the proposed method can greatly correct the cone-beam artifacts from clinical CBCT images reconstructed using 1/3 projections, and outperforms strong baseline methods both quantitatively and qualitatively.

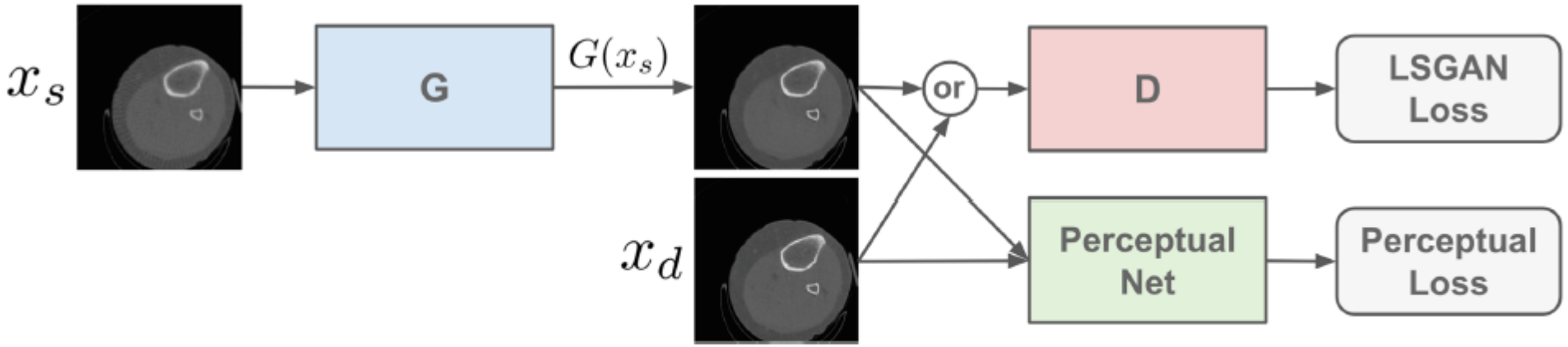


Fig. 1. The overall architecture of the proposed method.

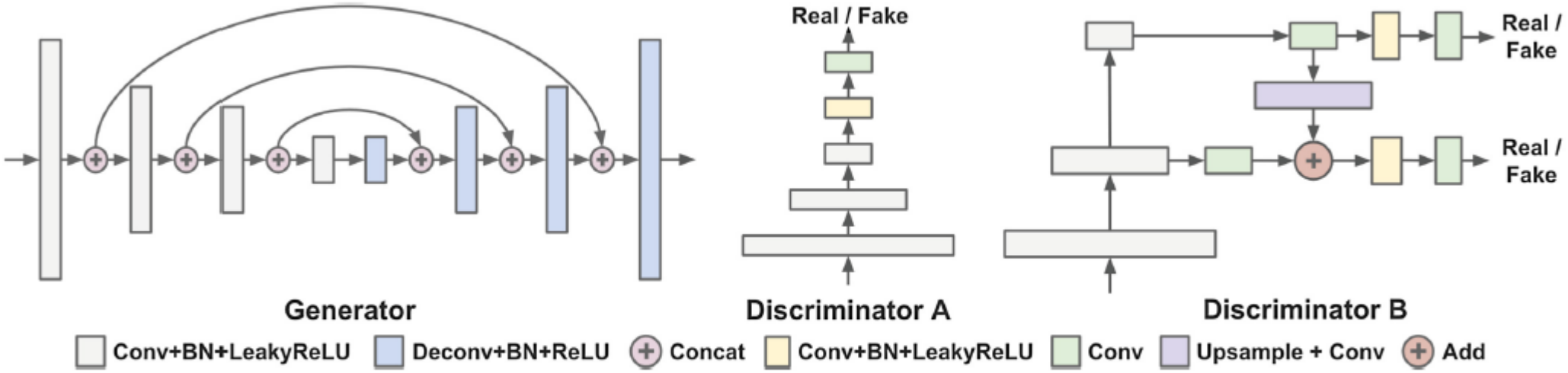


Fig. 2. Detailed network structure of the generator and discriminator.

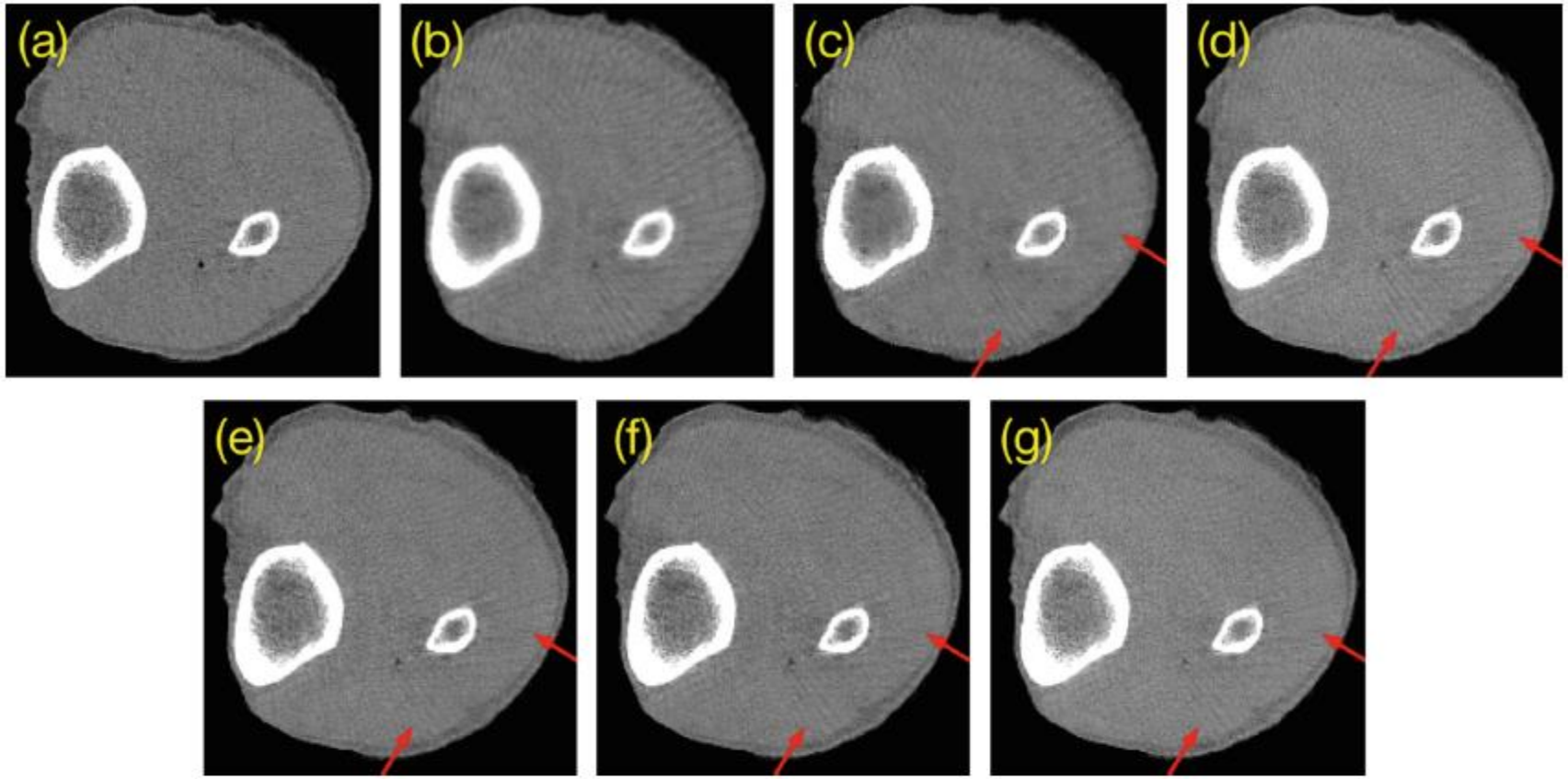


Fig. 4. Qualitative sparse-view CBCT artifact reduction results by different models. The same brightness and contrast enhancement are applied to the images for better and uniform visualization. (a) x_d (b) x_s (c) Baseline-MSE (d) Baseline-Perceptual (e) Ours-Focus (f) Ours-FPN (g) Ours-Focus+FPN

Localization and Labeling of Posterior Ribs in Chest Radiographs Using a CRF-regularized FCN with Local Refinement

Alexander Oliver Mader[1,2,3(✉)], Jens von Berg[3], Alexander Fabritz[2], Cristian Lorenz[3], and Carsten Meyer[1,2,3]

[1] Institute of Computer Science, Kiel University of Applied Sciences, Kiel, Germany
alexander.o.mader@fh-kiel.de
[2] Department of Computer Science, Faculty of Engineering, Kiel University, Kiel, Germany
[3] Department of Digital Imaging, Philips Research Hamburg, Hamburg, Germany

Abstract. Localization and labeling of posterior ribs in radiographs is an important task and a prerequisite for, e.g., quality assessment, image registration, and automated diagnosis. In this paper, we propose an automatic, general approach for localizing spatially correlated landmarks using a fully convolutional network (FCN) regularized by a conditional random field (CRF) and apply it to rib localization. A reduced CRF state space in form of localization hypotheses (generated by the FCN) is used to make CRF inference feasible, potentially missing correct locations. Thus, we propose a second CRF inference step searching for additional locations. To this end, we introduce a novel "refine" label in the first inference step. For "refine"-labeled nodes, small subgraphs are extracted and a second inference is performed on all image pixels. The approach is thoroughly evaluated on 642 images of the public Indiana chest X-ray collection, achieving a landmark localization rate of 94.6%.

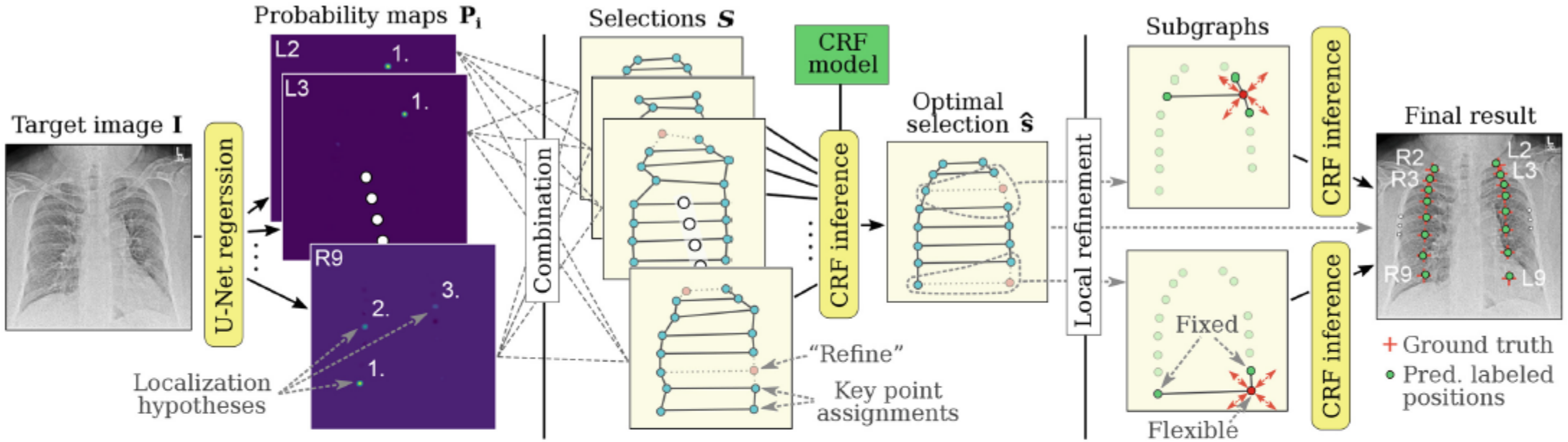
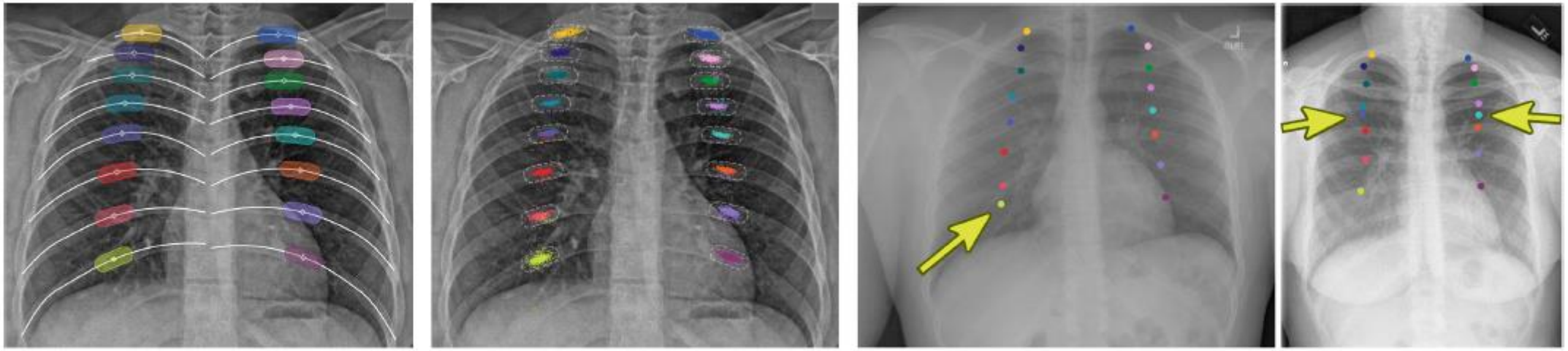


Fig. 1. Schematic illustration of our three-step approach: (a) Generation of localization hypotheses using a U-Net, (b) followed by a CRF modelling spatial relations between key points and (c) a final local refinement based on a subgraph considering the whole image domain.



(a) Annotations (b) Pred. positions (c) Typical errors

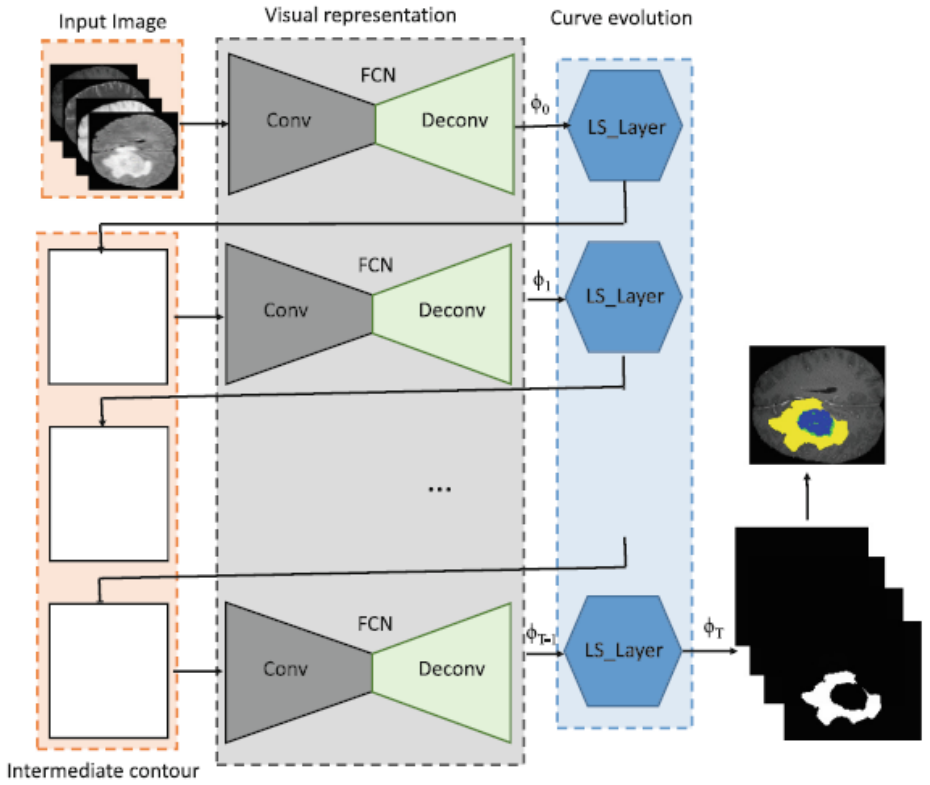
Fig. 3. (a) Illustration of the centerline and key point annotations and the resulting localization criterion, i.e., the area where a localization hypothesis is considered correct. Images in (a) and (b) are cropped and have been enhanced using adaptive histogram equalization. (b) Predicted positions in 642 test images visualized in a single image by registering the images using the true positions (affine and b-spline) and warping the predicted positions. Labels are shown color coded. (c) Typical errors involve an incorrect localization in the abdomen (first image) and chain errors caused by intermediate mistakes (second image).

Deep Recurrent Level Set for Segmenting Brain Tumors

T. Hoang Ngan Le(✉), Raajitha Gummadi(✉), and Marios Savvides(✉)

Carnegie Mellon University, Pittsburgh, USA
{thihoanl,rgummadi}@andrew.cmu.edu, msavvid@ri.cmu.edu

Abstract. Variational Level Set (VLS) has been a widely used method in medical segmentation. However, segmentation accuracy in the VLS method dramatically decreases when dealing with intervening factors such as lighting, shadows, colors, etc. Additionally, results are quite sensitive to initial settings and are highly dependent on the number of iterations. In order to address these limitations, the proposed method incorporates VLS into deep learning by defining a novel end-to-end trainable model called as Deep Recurrent Level Set (DRLS). The proposed DRLS consists of three layers, i.e., Convolutional layers, Deconvolutional layers with skip connections and LevelSet layers. Brain tumor segmentation is taken as an instant to illustrate the performance of the proposed DRLS. Convolutional layer learns visual representation of brain tumor at different scales. Brain tumors occupy a small portion of the image, thus, deconvolutional layers are designed with skip connections to obtain a high quality feature map. Level-Set Layer drives the contour towards the brain tumor. In each step, the Convolutional Layer is fed with the LevelSet map to obtain a brain tumor feature map. This in turn serves as input for the LevelSet layer in the next step. The experimental results have been obtained on BRATS2013, BRATS2015 and BRATS2017 datasets. The proposed DRLS model improves both computational time and segmentation accuracy when compared to the classic VLS-based method. Additionally, a fully end-to-end system DRLS achieves state-of-the-art segmentation on brain tumors.



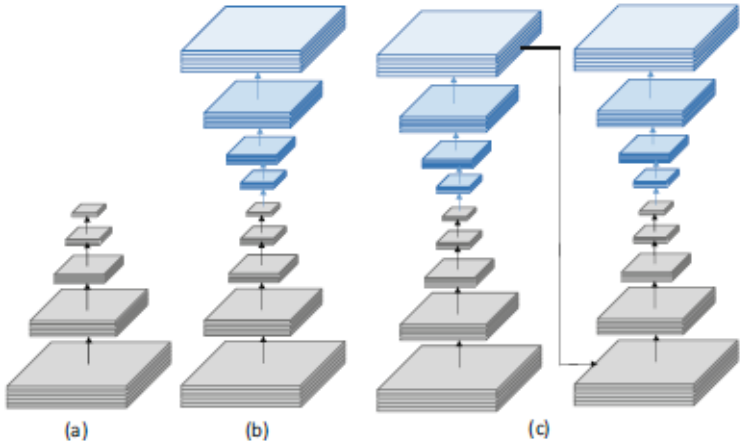
(a) The proposed DRLS with 2 main parts: visual representation and curve evolution

Algorithm 1 DRLS learning procedure

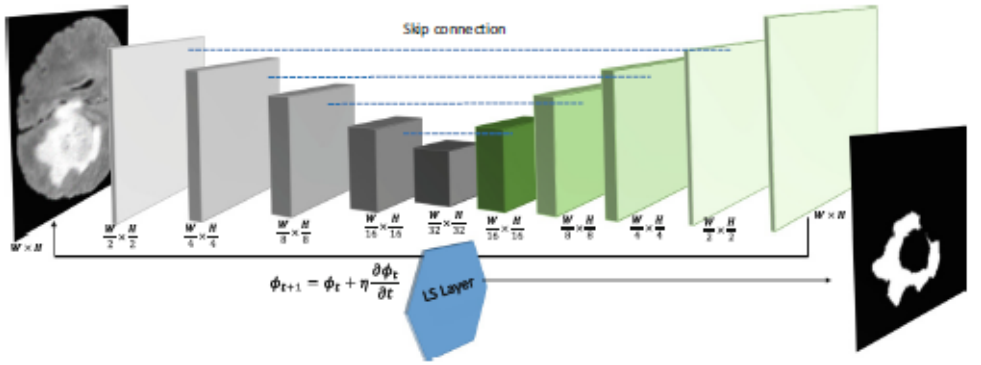
Input: Image **I**
Output: Segmenting result **S**
Procedure
visual representation by FCN
y = h(**I**)
Convert to [-0.5, 0.5]
$\phi = \xi(\mathbf{y}) - \xi(1 - \mathbf{y}) + \mathbf{y} - 0.5$
for $i = 0$ to $nsteps$ **do**
  $\phi_{old} = \phi$
  #LS curve evolution
  $\phi_{new} = \phi_{old} + \eta \frac{\partial \phi_{old}}{\partial t}$
  # visual representation by FCN
  $\mathbf{y} = h(\phi_{new})$
  # Convert to [-0.5, 0.5]
  $\phi = \xi(\mathbf{y}) - \xi(1 - \mathbf{y}) + \mathbf{y} - 0.5$
end for

(b) Psedo code of DRLS

Fig. 1. The proposed DRLS network and algorithm



(a) Comparison of different deep models. (1) CNN. (2) FCN. (3): RFCN



(b) VGG-16 with 5 conv. layer and 5 deconv. layers on which DRLS is built

Fig. 2. The details of the proposed DRLS

A Novel Bayesian Model Incorporating Deep Neural Network and Statistical Shape Model for Pancreas Segmentation

Jingting Ma[1(✉)], Feng Lin[1], Stefan Wesarg[3], and Marius Erdt[1,2]

School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore
{jma012,asflin,merdt}@ntu.edu.sg
Fraunhofer Singapore, Nanyang Technological University, Singapore, Singapore
[3] Visual Healthcare Technologies, Fraunhofer IGD, Darmstadt, Germany
stefan.wesarg@igd.fraunhofer.de

Abstract. Deep neural networks have achieved significant success in medical image segmentation in recent years. However, poor contrast to surrounding tissues and high flexibility of anatomical structure of the interest object are still challenges. On the other hand, statistical shape model based approaches have demonstrated promising performance on exploiting complex shape variabilities but they are sensitive to localization and initialization. This motivates us to leverage the rich shape priors learned from statistical shape models to improve the segmentation of deep neural networks. In this work, we propose a novel Bayesian model incorporating the segmentation results from both deep neural network and statistical shape model for segmentation. In evaluation, experiments are performed on 82 CT datasets of the challenging public NIH pancreas dataset. We report 85.32 % of the mean DSC that outperforms the state-of-the-art and approximately 12 % improvement from the predicted segment of deep neural network.

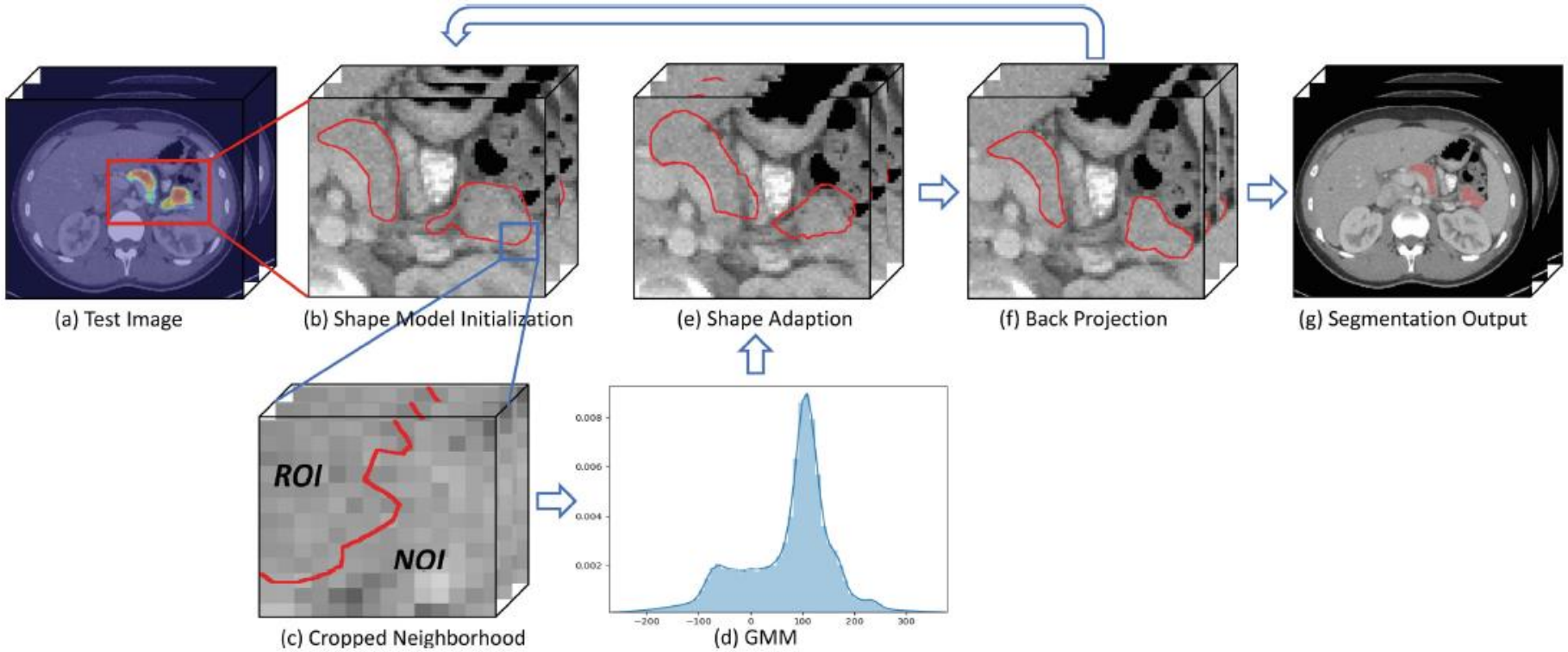


Fig. 1. This figure illustrates the pipeline of segmentation approach: given the test image with probability map (a), the shape model is initialized to fit the detected region (b); considering the neighborhood region around each landmark (c), a Gaussian Mixture Model is trained (d) to guide shape adaption (e); afterwards, project the shape onto statistical shape model (f); we obtain the segmentation output (g) when the convergence is reached.

Deep Attentional Features for Prostate Segmentation in Ultrasound

Yi Wang[1,2], Zijun Deng[3], Xiaowei Hu[4], Lei Zhu[4,5(✉)], Xin Yang[4], Xuemiao Xu[3], Pheng-Ann Heng[4], and Dong Ni[1,2]

[1] National-Regional Key Technology Engineering Laboratory for Medical Ultrasound, Guangdong Key Laboratory for Biomedical Measurements and Ultrasound Imaging, School of Biomedical Engineering, Health Science Cente Shenzhen University, Shenzhen, China
[2] Medical UltraSound Image Computing (MUSIC) Lab, Shenzhen, China
[3] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[4] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
lzhu@cse.cuhk.edu.hk
[5] Centre for Smart Health, School of Nursing, The Hong Kong Polytechnic University, Hong Kong, China

Abstract. Automatic prostate segmentation in transrectal ultrasound (TRUS) is of essential importance for image-guided prostate biopsy and treatment planning. However, developing such automatic solutions remains very challenging due to the ambiguous boundary and inhomogeneous intensity distribution of the prostate in TRUS. This paper develops a novel deep neural network equipped with deep attentional feature (DAF) modules for better prostate segmentation in TRUS by fully exploiting the complementary information encoded in different layers of the convolutional neural network (CNN). Our DAF utilizes the attention mechanism to selectively leverage the multi-level features integrated from different layers to refine the features at each individual layer, suppressing the non-prostate noise at shallow layers of the CNN and increasing more prostate details into features at deep layers. We evaluate the efficacy of the proposed network on challenging prostate TRUS images, and the experimental results demonstrate that our network outperforms state-of-the-art methods by a large margin.

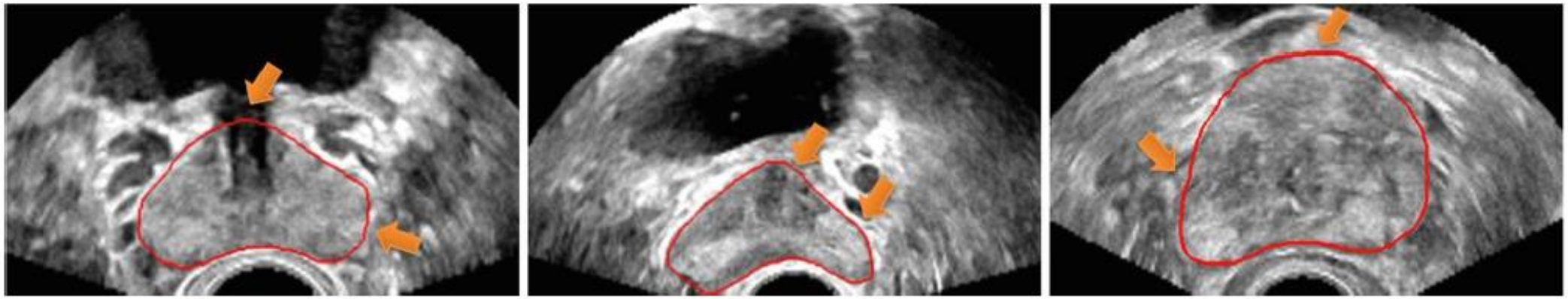


Fig. 1. Example TRUS images. Red contour denotes the prostate boundary. There are large prostate shape variations, and the prostate tissues present inhomogeneous intensity distributions. Orange arrows indicate missing/ambiguous boundaries.

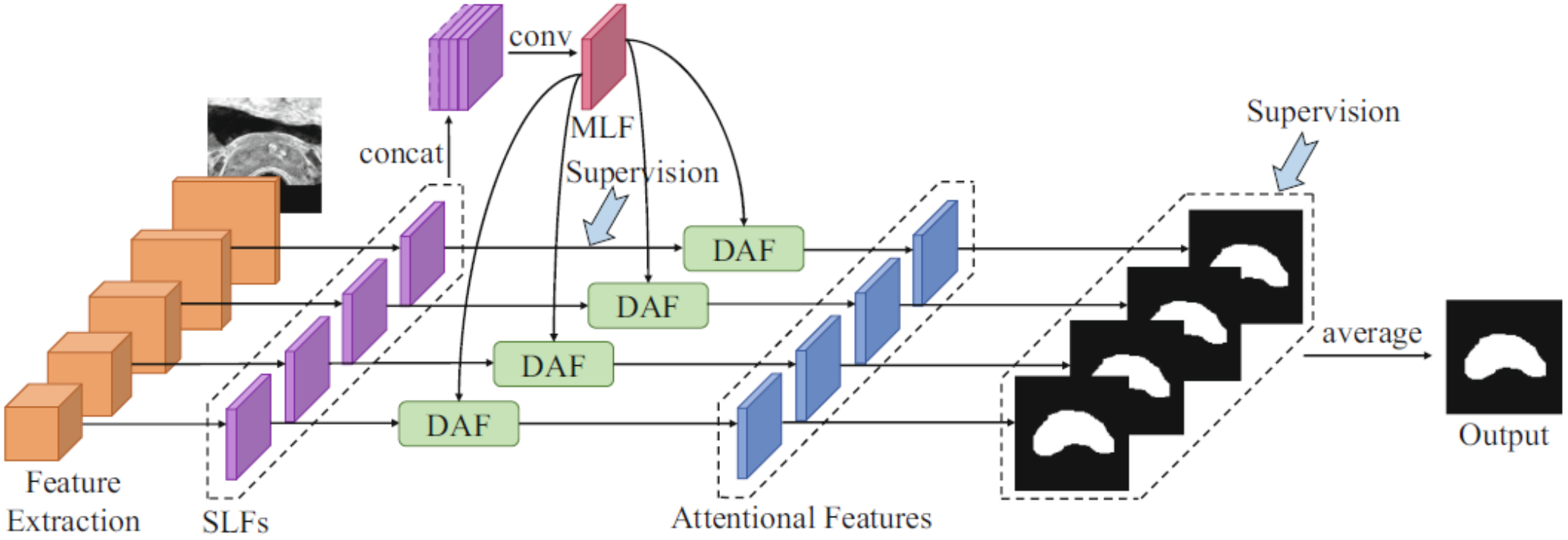


Fig. 2. The schematic illustration of our prostate segmentation network with deep attentional features (DAF). SLF: single-layer features; MLF: multi-layer features.

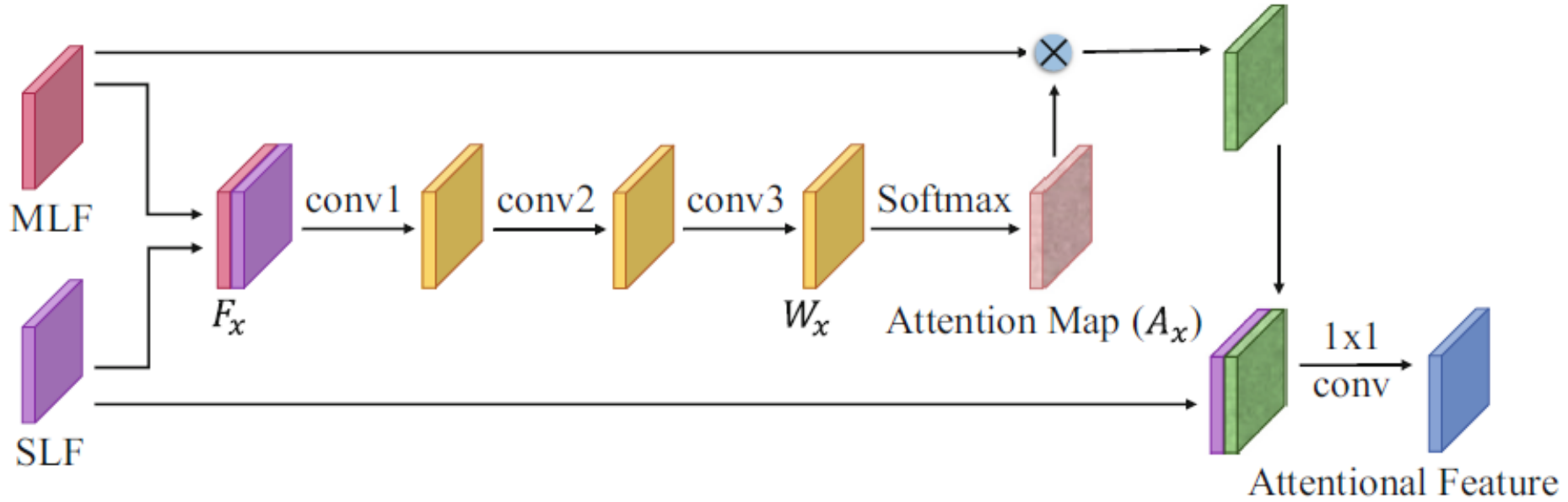


Fig. 3. The schematic illustration of the *deep attentional feature* (*DAF*) module.

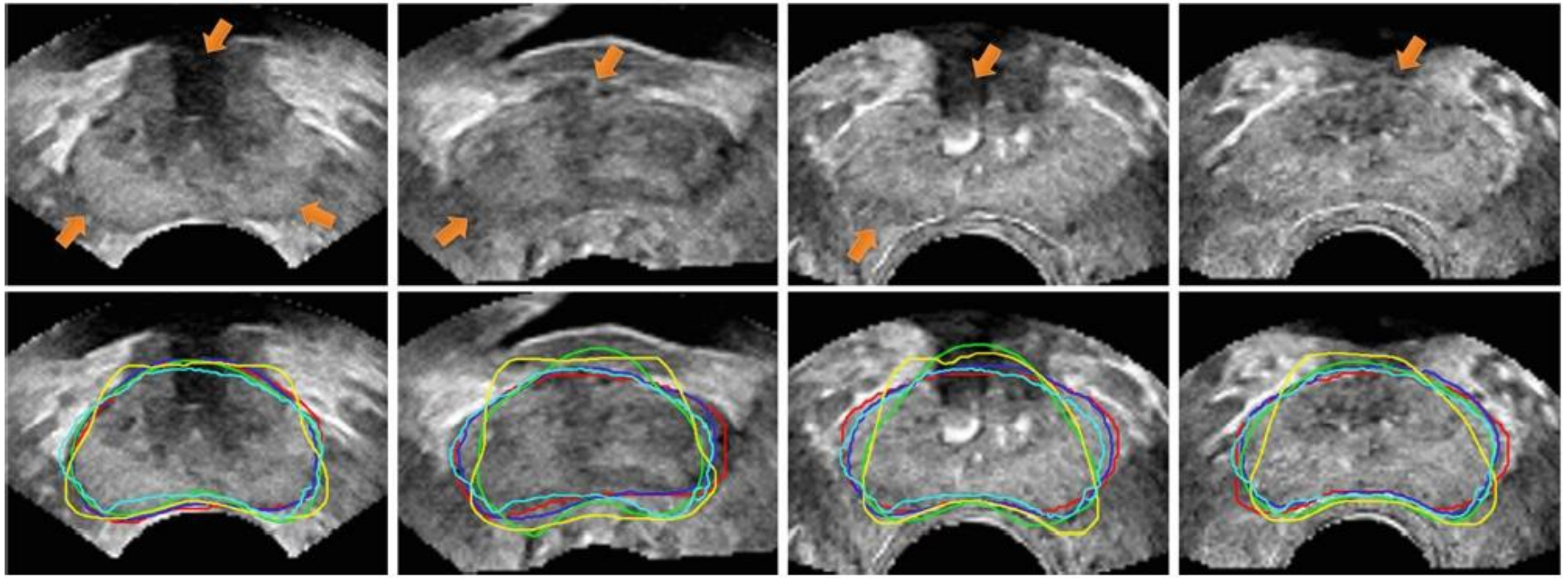


Fig. 4. Visual comparison of prostate segmentation results. Top row: prostate TRUS images with orange arrows indicating missing/ambiguous boundaries; bottom row: corresponding segmentations from our method (blue), U-Net (cyan), BCRNN (green) and FCN (yellow), respectively. Red contours are ground truths. Our method has the most similar segmented boundaries to the ground truth.