□ Assignment 1: Class Activation Mapping from Pre-Trained Networks 11 of 21 points (52%)

Please implement all coding parts in the notebook that you can download here: https://seafile.ifi.uzh.ch/f/9835a05b31034a51a540/?dl=1

This assignment is concerned with the implementation of a Class Activation Mapping (CAM) technique in pytorch. Particularly, we implement the Grad-CAM technique as introduced in the lecture. For Grad-CAM, we need to compute the partial derivative of the logit output z_o of the network for class o with respect to the output $A \in \mathbb{R}^{Q \times K \times M}$ of the last convolution map, and average this over the spatial dimensions of the feature map:

$$a_q^o = \frac{1}{KM} \sum_{k,m} \frac{\partial z_o}{\partial a_{q,k,m}}$$

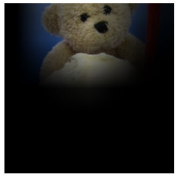
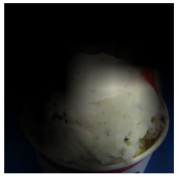
The visualization map $\mathbf{M}_o \in \mathbb{R}^{K \times M}$ for class o is computed as the weighted sum of the feature map, where negative values are clipped to 0:

$$\mathbf{M}_o = \max\left(\sum_q a_q^o \mathbf{A}_q, 0\right)$$

which is normalized to $[0, 1]$ by dividing by its maximum value:

$$\mathbf{M}_o' = \frac{\mathbf{M}_o}{\max \mathbf{M}_o}$$

Finally, we want to apply this activation map by multiplying it to the pixels of the original image, in order to remove unimportant pixels from the image. An exemplary image and the activations for two different classes (Teddy and Ice Cream) can be found below.

We apply the Grad-CAM technique to a pre-trained ResNet-18 topology, which we need to adapt such that it returns all information that is required to perform the task. Therefore, we need to analyze the implementation of the ResNet topology, particularly the implementation of the *forward* function, which can be found in its code and is given below. Particularly, the layers *layer1* -- *layer4* are outputs of the residual convolution blocks, *avgpool* is implemented as a Global Average Pooling, and *fc* is the final fully-connected layer $\mathbf{W} \in \mathbb{R}^{O \times Q}$ that transforms the output of dimension $h^{\rightarrow} \in \mathbb{R}^{Q}$ to the final $O = 1000$ logits that represent the 1000 ImageNet classes. Note that this weight matrix is stored as *fc.weight* as provided in its code.

1(a): (text question) Image Preprocessing

Status	Answered	
Your score	2 / 2	100%

Response

Our input color image can be of any arbitrary size. Describe the steps that are required so that the image can be used with the pre-trained ResNet-18 network.

16 words

Solution

1(b): (text question) Grad-CAM Derivative

Status	Answered	
Your score	0 / 5	0%

Response

Given the network topology of ResNet-18 and the terminology as described above, analytically compute the partial derivative of the logit z_o with respect to the feature map $a_{q,k,m}$. Compute the weight a_q^o as defined above. Provide all steps for the computation.

3 words

Solution

□ 1(c): (code question) Network Instantiation

Status	Answered	
Your score	1.5 / 3	50%

Comment / assessment

Feature map not saved and returned

Response

Write a code block that instantiates and downloads the pre-trained ResNet-18 network using functionality in *torchvision*. Adapt its forward function to return whatever you require for computing the Grad-CAM as theoretically analyzed in (b).

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 1(d): (code question) Feature Extraction

Status	Answered	
Your score	3 / 3	100%

Response

Implement the preprocessing steps as defined in (a). Extract all required elements using the pre-trained ResNet-18 to compute the visualization.

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

1(e): (code question) Visualization and Overlay

Status	Answered	
Your score	4.5 / 8	56%

Comment / assessment

Incorrect gradient.

Incorrect dimension of map to multiply with image.

Need to apply map to original (unnormalized) image

Response

Implement a function that computes the visualization map $\mathbf{M}'_o$ as defined in above for a given class o and applies this element-wise to the image.

- The implementation is included in the Jupyter notebook uploaded at the end of this section.

Solution

1(f): Jupyter Notebook Upload

Status	Answered	
Your score	0 / 0	0%

Response

After all **implementations of Assignment 1** are finished, please upload **the Jupyter notebook** here. The upload of partial implementations is also encouraged. **Other ways of submissions of the solution (e.g. by email) are not allowed and will not be counted.** Other documents (e.g. automatically downloaded data or figures generated with the notebook) are not required to be uploaded.

Do not forget to upload the notebook. Without this file, tasks 1(c), 1(d) and 1(e) will not receive any points.

File

Solution

go back to overview

Assignment 2: Classification of the Gender from the First Name 15 of 26 points (58%)

Please implement all coding parts in the notebook that you can download here: https://seafile.ifi.uzh.ch/f/3e3332faaaee4b9b8579/?dl=1

We want to train an Elman network that can predict the gender of a person from their first name. The data that we will make use of is given online in the UCI data repository: https://archive.ics.uci.edu/dataset/591/gender+by+name

For this purpose, we need to encode the name of the person such that it can be understood by the network. Particularly, we transform all upper-case characters to lower-case, and encode all lower-case characters, including a '-' character, into a one-hot encoding of size $x^{\{s\}} \in \mathbb{R}^D$. A name consists of a sequence of S characters, such that each name is represented as $\mathbf{X} \in \mathbb{R}^{S \times D}$.

We define the Elman network to process such input by the following equations:

$$\vec{h}^{\{s\}} = g\left(\mathbf{W}^{(1)} x^{\{s\}} + \mathbf{W}^{(r)} \vec{h}^{\{s-1\}}\right)$$

$$z^{\{s\}} = \mathbf{W}^{(2)} \vec{h}^{\{s\}}$$

where $\mathbf{W}^{(1)}$, $\mathbf{W}^{(r)}$ and $\mathbf{W}^{(2)}$ are learnable matrices, $g(\cdot)$ is an activation function, and $\vec{h}^{\{s\}}$ and $z^{\rightarrow\{s\}}$ are, respectively, the hidden representation and the logit output of the network for sequence element $x^{\rightarrow\{s\}}$. Finally, a prediction whether the name is male or female is made by interpreting the output of the network.

2(a): (text question) Network Design

Status	Answered	
Your score	0 / 2	0%

Response

What are the dimensionalities of the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(r)}$? What is an appropriate activation function $g(\cdot)$ and why? How many inputs and outputs does the second layer need?

10 words

Solution

2(b): (text question) Network Output

Status	Answered	
Your score	1 / 3	33%

Comment / assessment

only loss is correct

Response

For the task at hand, which sequence information do you compute/store for the final prediction? How can you decide whether the name is male or female, based on the output of the network? Which loss function do you need to train the network?

20 words

Solution

□ 2(c): (code question) Character Encoding

Status	Answered	
Your score	2 / 3	67%

Comment / assessment

-1: D is wrong

-1:no function/dict returned/stored

Response

Implement a function or a data structure that provides an encoding for a given character *c* that can be contained in a name.

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 2(d): (code question) Dataset Implementation

Status	Answered	
Your score	1 / 5	20%

Comment / assessment

only __len__is correct (+1)

Response

Implement a *torch.utils.data.Dataset* class, which provides the encoded names. Assure that all encodings have the same sequence length S by applying a reasonable padding.

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 2(e): (code question) Elman Network Implementation

Status	Answered	
Your score	6 / 6	100%

Response

Implement the Elman network as designed in (a) as a *torch.nn.Module*. Remember that inputs $X \in \mathbb{R}^{B \times S \times D}$ will be processed in batches.

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 2(f): (code question) Network Training

Status	Answered	
Your score	3 / 5	60%

Comment / assessment

-0.5: no defined K or C(output)

-0.5: loss is not consistent with task(b)

-0.5: dataset initialization is wrong

-0.5: normalization of accuracy is wrong

Response

Train your network with the loss function discussed in (b). Report the training set accuracy after each epoch, using the evaluation criterion defined in (b).

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 2(g): (text question) Network Topologies

Status	Answered	
Your score	2 / 2	100%

Response

The classification accuracy of the network is not high enough, so you decide to make use of a different network to solve your task. Name two conceptually different options of network topologies that can be used with sequential data.

6 words

□ Solution

□ 2(h): Jupyter Notebook Upload

Status	Answered	
Your score	0 / 0	0%

Response

After all **implementations of Assignment 2** are finished, please upload **the Jupyter notebook** here. The upload of partial implementations is also encouraged. **Other ways of submissions of the solution (e.g. by email) are not allowed and will not be counted.** Other documents (e.g. automatically downloaded data or figures generated with the notebook) are not required to be uploaded.

Do not forget to upload the notebook. Without this file, tasks 2(c), 2(d) , 2(e) and 2(f) will not receive any points.

File

□ Solution

□ go back to overview

□ Assignment 3: House Number Image Classification 20.5 of 27 points (76%)

Please implement all coding parts in the notebook that you can download here: https://seafile.ifi.uzh.ch/f/af8d5ac39a104622b106/?dl=1

A convolutional network should be designed for the automatic classification of small images from the Street View House Numbers (SVHN) dataset, which is composed of 10 different classes, one per digit. An implementation of the dataset can be found in *torchvision.datasets*. The original color images are of size 32×32 pixels, examples can be found below.

Since the images are embedded by other digits on both sides, you decide that to exclude the horizontal padding, but only pad vertically. You implement the network with the following layers:

- A convolution layer with kernel size 5×5, 32 output channels, stride 1, padding 2 in vertical dimension only.
- A maximum pooling layer of size 2×2.
- A convolution layer with kernel size 5×5, 64 output channels, stride 1, padding 2 in vertical dimension only.
- A maximum pooling layer of size 2×2.
- A convolution layer with kernel size 5×5, 128 output channels, stride 1, padding 2 in vertical dimension only.

□ 3(a): (text question) Network Parameters

Status	Answered	
Your score	1 / 3	33%

Comment / assessment

-1: wrong number of FC

-1: wrong computation of conv layer

Response

What is the size of the feature map that represents the output of the last convolutional layer? Provide the details of your computation. What is the dimensionality $O \times K$ of the final fully-connected layer?

10 words

□ Solution

3(b): (text question) Regularization

Status	Answered	
Your score	3 / 4	75%

Comment / assessment

-1: missing reasonable explanation

Response

Regularization plays an important role in deep learning. Name three conceptually different regularization techniques. Explain one of them in more detail.

9 words

Solution

3(c): (code question) Dataset and Data Loaders

Status	Answered	
Your score	3.5 / 4	88%

Comment / assessment

-0.5: images are upscaled to a too large size

Response

Instantiate the training and test set of the SVHN dataset from *torchvision*. Set the dataset to automatically download. Apply appropriate transforms. Instantiate appropriate data loaders.

The implementation is included in the Jupyter notebook uploaded at the end of this section.

Solution

□ 3(d): (code question) Convolutional Network Implementation

Status	Answered		
Your score	3.5 / 5		70%

Comment / assessment

-1: missing activation function

-0.5: wrong size for FC

Response

Implement and instantiate the convolutional network in *pytorch*. Include at least one regularization technique from (b). Are there any kinds of layers that you additionally need to include?

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 3(e): (code question) Network Training

Status	Answered		
Your score	4.5 / 6		75%

Comment / assessment

-0.5: wrong loss function

-1: missing computation of test_acc

Response

Instantiate an appropriate loss function and an optimizer. Train the network for 10 epochs on the GPU. Compute the average training set loss within an epoch. Compute the test set accuracy at the end of each epoch.

□ The implementation is included in the Jupyter notebook uploaded at the end of this section.

□ Solution

□ 3(f): (text question) Improve Results

Status	Answered		
Your score	3 / 3	[redacted]	100%

Response

After training, you realize that the accuracy on the test set is not high enough. Discuss three possible techniques to improve the results.

9 words

□ Solution

□ 3(g): (text question) Non-Digit Input Image

Status	Answered		
Your score	2 / 2	[redacted]	100%

Response

When classifying house numbers from real street views, it might be possible that the network is presented with an image that does not show a digit of a house number and, thus, does not belong to any of the classes. How do deep networks usually react to such inputs? How can this situation be improved? Describe one method in detail.

32 words

□ Solution

□ 3(h): **J**upyter Notebook Upload

Status	Answered		
Your score	0 / 0		0%

Response

After all **implementations of Assignment 3** are finished, please upload **the Jupyter notebook** here. The upload of partial implementations is also encouraged. **Other ways of submissions of the solution (e.g. by email) are not allowed and will not be counted.** Other documents (e.g. automatically downloaded data or figures generated with the notebook) are not required to be uploaded.

Do not forget to upload the notebook. Without this file, tasks 3(c), 3(d) and 3(e) will not receive any points.

File

□ Solution

□ go back to overview

□ Assignment 4: True/False Questions 14.5 of 22 points (66%)

In this assignment, questions about several topics from the lecture are presented. Always select all answers that are correct.

Please be aware that 1 point is awarded for each correct answer, and **0.5 points are deducted for each incorrect answer**. If no answer is given, 0 points will be awarded. Any of the four questions cannot end up with a negative overall score.

□ 4(a): (true/false question) Partial Derivatives

Status	Answered	
Your score	3.5 / 5	70%

Response

Given the given input sample $\vec{x}$ with one-hot-encoded target vector $\vec{t}$, as well as the following two-layer network and loss function, which of the partial derivatives are correct?

$\vec{a} = \mathbf{W}^{(1)} \cdot \vec{x}$

$\vec{h} = \tanh(\vec{a})$

$\vec{z} = \mathbf{W}^{(2)} \cdot \vec{h}$

$\forall o : y_o = \frac{e^{z_o}}{\sum_{o'} e^{z_{o'}}}$

$J = -\sum_o t_o \cdot \log y_o$

Unanswered	Right	Wrong	
□	□	□	$\frac{\partial \vec{h}}{\partial \vec{a}} = \vec{h} \odot (1 - \vec{h})$ (where $\odot$ represents the element-wise multiplication operation)
□	□	□	$\frac{\partial \vec{z}}{\partial \vec{h}} = \mathbf{W}^{(2)}$
□	□	□	$\frac{\partial J}{\partial y_o} = y_o - t_o$
□	□	□	$\frac{\partial z_o}{w^{(2)}_{o,k}} = h_k$
□	□	□	$\frac{\partial a_k}{\partial x_d} = w^{(1)}_{k,d}$

□ Solution

□ 4(b): (true/false question) Vanishing Gradients

Status	Answered	
Your score	3.5 / 5	70%

Response

Which of the following techniques or interventions can be used to fight against the vanishing gradient problem?

Unanswered	Right	Wrong	
□	□	□	Add residual connections connections in your network design.
□	□	□	Change all activation functions in between layers to *ReLU*.
□	□	□	Add weight decay in the optimizer.
□	□	□	Reduce the learning rate during training.
□	□	□	Apply batch normalization between layers.

□ Solution

□ 4(c): (true/false question) Convolutional Classification Networks

Status	Answered	
Your score	4.5 / 6	75%

Response

Please mark all correct and incorrect statements about convolutional, residual and classification networks.

Unanswered	Right	Wrong	
□	□	□	Decreasing all inputs to SoftMax by a constant value decreases all output probabilities.
□	□	□	Convolutional layers with kernel size 5×5 have 25 learnable parameters.
□	□	□	Adding Residual Connections increases the number of learnable parameters of the network.
□	□	□	Global Average Pooling allows to use inputs of various resolutions.
□	□	□	After applying SoftMax, the sum over all probabilities is one: $\sum_o y_o = 1$
□	□	□	Deformable Convolution can be used to increase the size of the receptive field of the convolution operation.

□ Solution

4(d): (true/false question) Various Questions

Status	Answered	
Your score	3 / 6	50%

Response

Please select all correct and wrong statements for the questions below, which regard various topics handled during the lecture.

Unanswered	Right	Wrong	
☐	☐	☐	Auto-encoder networks require labeled training data.
☐	☐	☐	Gradient-based Adversarial Attacks compute the partial derivative of the loss with respect to the input.
☐	☐	☐	The Adapted SoftMax loss used for open-set classification aims at reducing the probabilities for all known classes to 0, when presented with a negative training sample that belongs to no known class.
☐	☐	☐	Data Augmentation is used to reduce the number of epochs required for training.
☐	☐	☐	Generative Adversarial Networks require noise inputs to produce different samples.
☐	☐	☐	Batch Normalization removes the necessity to add a bias neuron in the following layer.

Solution

There are no more attempts at your disposal.