Università
della
Svizzera
italiana

Faculty
of Informatics

Image and Video Processing (Spring 2024)

Assignment 5: Deep Learning in Image Processing May 21, 2024

This assignment is more structured than the previous one. For each exercise, you have a dedicated Jupyter Notebook where you must fill in the code. You need some basic knowledge of Machine Learning and PyTorch for this assignment; if you are not familiar with one, try to team up with someone more confident in these topics; if you can't find someone, don't worry, you can find a lot of articles on these topics.

1 Image classification [7 points]

Using convolutions, we can filter images in different ways, in neural networks, we use convolutions to extract features from images. We can learn specific convolutional filters such that the output of the convolutional layer is a feature map that highlights specific features in the input image. This is done by learning the weights of the convolutional filters during training. We can apply this principle to the image classification task using a convolutional neural network (CNN). Your task is implementing a simple CNN for image classification using PyTorch and training it on the CIFAR-10 dataset.

1.1 Build the model [3 points]

Task. Following the model's description in Table 1, implement the model using PyTorch.

Layer type	Parameters
Conv2d	Channels: 32, Kernel Size: 3, Padding: 1
ReLU	
MaxPool2d	Kernel Size: 2
Conv2d	Channels: 64, Kernel Size: 3, Padding: 1
ReLU	
MaxPool2d	Kernel Size: 2
Conv2d	Channels: 128, Kernel Size: 3, Padding: 1
ReLU	
MaxPool2d	Kernel Size: 2
Flatten	
Linear	Input: $128 \cdot 4 \cdot 4$, Output: 128
ReLU	
Linear	Input: 128, Output: 10

Table 1: Model structure

1.2 Model training [2 points]

To train a model, we need an objective function to minimize; in this case, the objective function is the Cross-Entropy Loss defined as:

$$\text{Cross-Entropy}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{1}$$

where y_i is the true label and $\hat{y}_i$ is the predicted label. The values are expected to be in the range $[0, 1]$, so you should apply a softmax activation function to the output of the model, defined as:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2}$$

Note. When using the Cross-Entropy Loss function in PyTorch, you are not required to apply a softmax activation function to the output of the model, as the Cross-Entropy Loss function applies the softmax function internally.

Task. Evaluate the model on the test set and report the accuracy.

1.3 Data augmentation [2 points]

To improve the performance of the model, you can apply data augmentation. Data augmentation is a technique for increasing the size of the training set by applying transformations to the input data. This can help the model generalize the unseen data better.

Task. Train the model with the augmented data and evaluate the model on the test set. Implement data augmentation using the following transformations:

- Horizontal flip
- Vertical flip
- Rotation
- Change the brightness, contrast, saturation, and hue of the image (Color Jitter)

Hint. You can use the torchvision.transforms module to apply the transformations.

Note. When you apply data augmentation, you usually need more time to train the model until it converges.

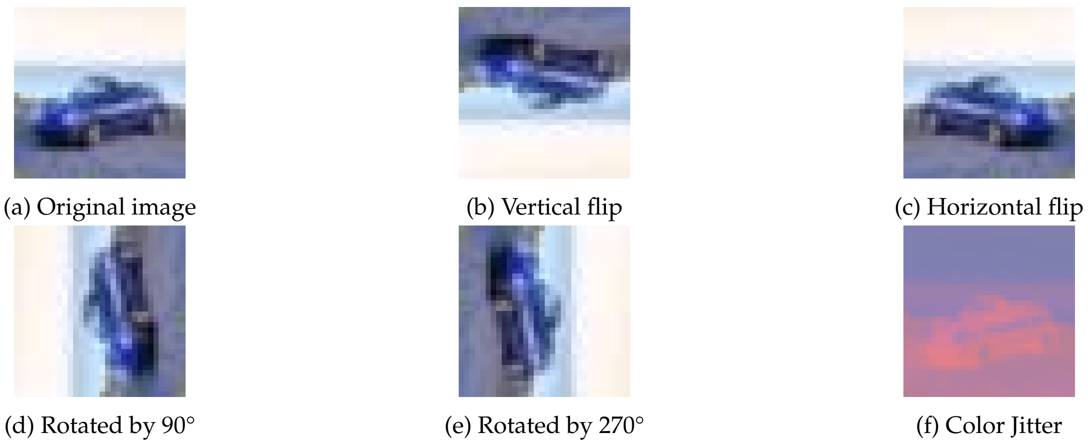
(a) Original image (b) Vertical flip (c) Horizontal flip
(d) Rotated by 90° (e) Rotated by 270° (f) Color Jitter

Figure 1: Expected results from the different transformations

2 Image denoising [8 points]

We can use an autoencoder to learn a compressed representation of the input data. The autoencoder consists of an encoder and a decoder. The encoder maps the input data to a lower-dimensional space, and the decoder maps the lower-dimensional representation back to the original input space. The autoencoder is trained to minimize the reconstruction error, which is the difference between the input data and the output of the decoder. We can use an autoencoder to denoise images by training the autoencoder on noisy images and the corresponding clean images.
The mathematical formulation of the autoencoder is given by:

$$\hat{x} = \text{Decoder}(\text{Encoder}(x)) \tag{1}$$

where x is the input data, $\hat{x}$ is the decoder output, and Encoder and Decoder are the encoder and decoder functions, respectively.
The reconstruction error is defined as:

$$\text{MSE}(x, \hat{x}) = \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \tag{2}$$

2.1 Peak signal-to-noise ratio (PSNR) metrics [1 point]

The Peak signal-to-noise ratio (PSNR) is defined as:

$$\text{PSNR}(x, \hat{x}) = 10 \cdot \log_{10} \left(\frac{\text{MAX}_\text{I}{}^2}{\text{MSE}(x, \hat{x})} \right) \tag{3}$$

Task. As a metric for evaluating the already defined model, implement the PSNR metrics for the images in the test set.

2.2 Structural similarity index measure (SSIM) [3 points]

The Structural similarity index measure (SSIM) is defined as:

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)} \tag{4}$$

The SSIM is computed over a window in the image x and $\hat{x}$. The mean is calculated as:

$$\mu_x = (x * g) \tag{5}$$

where g is a circular symmetric Gaussian filter with a size corresponding to the window size. The covariance is calculated as:

$$\sigma_{x,y} = ((x \cdot y) * g) - (\mu_x \cdot \mu_y) \tag{6}$$

Task. As a second metric for evaluating the model, implement the SSIM for the images in the test set. Use a window size of 11 and a standard deviation of 1.5 for the Gaussian filter. For the constants C_1 and C_2 use $C_1 = 0.01^2$ and $C_2 = 0.03^2$.

2.3 Model training and evaluation [2 points]

Train an autoencoder to denoise images using the FashionMNIST dataset. Report the PSNR and SSIM metrics for the images in the test set. Report an image of the denoising process for a sample image in the test set at the beginning, middle, and end of the training.

2.4 Modify the predefined model [2 points]

In the original model the decoder is implemented using Conv2d and Upsample layers. Implement the decoder using ConvTranspose2d layers and train the model. Report the PSNR and SSIM metrics for the images in the test set. Report an image of the denoising process for a sample image in the test set at the beginning, middle, and end of the training.

3 Image generation [5 points]

We can use a variational autoencoder (VAE) to learn a probabilistic representation of the input data. This allows us to generate new data samples by sampling from the learned distribution. The VAE consists of an encoder and a decoder. The encoder maps the input data to a lower-dimensional space, and the decoder maps the lower-dimensional representation back to the original input space. The encoder outputs the mean and variance of the learned distribution, and the decoder samples from this distribution to generate new data samples. The VAE is trained to minimize the reconstruction error and the Kullback-Leibler (KL) divergence between the learned and prior distributions.
The mathematical formulation of the VAE is given by:

$$\mu, \sigma = \text{Encoder}(x) \tag{1}$$

$$z \sim q(z|x) = \mathcal{N}(\mu, \sigma) \tag{2}$$

$$\hat{x} = \text{Decoder}(z) \tag{3}$$

where x is the input data, μ and σ are the mean and variance of the learned distribution, z is the sampled representation, and $\hat{x}$ is the output of the decoder.
Then if we want to generate a new sample the generation process is defined as:

$$z' \sim \mathcal{N}(0, I) \tag{4}$$

$$\hat{x}' = \text{Decoder}(z') \tag{5}$$

The reconstruction error is defined as:

$$\text{MSE}(x, \hat{x}) = \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \tag{6}$$

The KL divergence is defined as:

$$\text{KL}(q(z|x)||p(z)) = -\frac{1}{2} \sum_{i=1}^{N} (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \tag{7}$$

The final objective function is defined as:

$$\text{Loss}(x, \hat{x}, \mu, \sigma) = \text{MSE}(x, \hat{x}) + \text{KL}(q(z|x)||p(z)) \tag{8}$$

3.1 Model training and evaluation [2 points]

Train the defined VAE to generate new images using the MNIST dataset. Report PSNR and SSIM metrics for the images in the test set. Generate 10 new images by sampling from the learned distribution and report the images.

3.2 Conditional image generation [3 points]

We don't have any control over the generated images, we can't generate images of a specific digit. We can use a conditional variational autoencoder (CVAE) to address this issue. The CVAE is an extension of the VAE that conditions the generation process on a specific class label. This allows us to generate images of a specific digit by providing the class label as input to the decoder. We can generate images of a specific digit by adding label information in the latent space to force a deterministic constraint; this can be done by applying a linear projection on the class label and adding it to the latent space. This process can be defined as:

$$z = z + \text{Linear Projection(label)} \tag{9}$$

Task. Modify the VAE into a CVAE. Train a CVAE to generate new images of a specified digit using the MNIST dataset. Report PSNR and SSIM metrics for the images in the test set. Generate 5 images for 3 digits of your choice and report the images.

Submission

You should submit one ZIP-file via iCorsi containing:

- All your code in Python appropriately commented, the produced pictures that you obtained, and the model's weights so that we can reproduce your results. **Failure to make your code reproducible will result in deducted points.**
- A complete PDF report detailing your solution and partial results for each exercise (Maximum 7 pages). **Points will be deducted for reports exceeding the limit.**

Grading will be **solely** based on the provided PDF report so we encourage clarity and detailed answers. We recommend using LaTeX or Overleaf to write the report. Usage of ChatGPT or any other natural language model is strictly prohibited and will be severely punished.

Solutions must be returned on June 5, 2024 via iCorsi3