Deep Learning KU (DAT302.UF) - Winter Semester 2024

Deep Learning: Course Projects [40 pts]

This document consists of a list of project topic suggestions that your group can choose one from. The goal of the course project is to gain a practical *and* theoretical understanding of modern deep learning project pipelines. The suggested topics cover a variety of modern deep learning architectures and applications. After picking a topic, you should decide on the dataset you want to use.

Overview: Your group will work on **exactly one of the topics** from the *project topics* list below. To this end, pick **exactly one dataset** that fits the topic description. After implementing and training the models for the specific dataset/task, you are expected to present a structured description of the project as a report. Finally, at the end of the semester, you will present your work to the other students of this class.

General Remarks: Using `PyTorch`, implement the model of your choice **from scratch**: That is, do not use packages or libraries that already implement the model for you (e.g., `diffusers` for Diffusion Models). One of this project's goals is to become familiar with the practical implementation of such model architectures. If you use any resources, make sure to cite them properly (code and report).

Project Reports: Submitted reports should include the following sections:

1. **Introduction**: Describe the problem you want to solve and the dataset you will use
2. **Methods**: Describe the model architecture in detail. Write down a mathematical definition of the loss function used for optimization. Thoroughly explain the different parts of the loss function and briefly discuss why it makes sense to use the loss function that you optimize.
3. **Results**: Elaborate on the model training, selection and evaluation process. Evaluate your model and clearly present the results.
4. **Discussion and Conclusions**: Describe the main observations. How well does the model work? Also discuss the evaluation method(s) of your choice. Briefly outline the limitations/shortcomings. What are the conclusions?

Include a clear description of your final architecture (e.g., regularization approaches, convolutional layer specifications, activations, latent dimensions, and other hyperparameters). Report your model training details (e.g., loss function and optimization), and the amount of parameters in your networks. Provide tables that depicts your hyperparameter search, as well as plots of training, validation and test set losses across training iterations.

Datasets: Topics suggested below (except the topic [RNN]) are mainly suited for image datasets. The images in the dataset you choose **must be color images**, i.e., have at least three channels (RGB). Do not use grayscale images. Your project **can use any RGB colored image dataset of your choice** as long as the computational demand is surmountable on your end.

Some dataset options: CelebA[1], STL-10[2], Caltech Football Numbers[3], SVHN[4], CIFAR-10/100[5]. You can also create and use a colored version of any grayscale image dataset (e.g., Wildlife MNIST[6]). This can also be implemented by (reasonably) binarizing any grayscale dataset based on a pixel value threshold, and then assigning random colors to the white parts and keeping the background black (or differently colored). Some grayscale image dataset options for this purpose: Caltech 101 Silhouettes[7] (you can use a subset of the 101 categories), EMNIST[8], handwritten character datasets in different languages such as Chinese MNIST[9], Kuzushiji-MNIST[10] or Kannada-MNIST[11].

Project Topics

1) [DiffModel] Diffusion Models

Your task will be to train a diffusion model $p_\theta(\mathbf{x})$ on an image dataset. Use U-Net-like convolutional network architectures that also receive the time t as an embedding. Mathematically define the problem that the diffusion model is trying to solve and show how you can sample from such a model using an iterative algorithm.

Following a detailed model selection process, implement the sampling algorithm and present your generated data samples. Investigate the influence of several training hyperparameters and sampling hyperparameters.

For reference:

- Ho et al., "Denoising Diffusion Probabilistic Models", NeurIPS 2020
- Song and Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", NeurIPS 2019

[1] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[2] https://cs.stanford.edu/~acoates/stl10/
[3] https://github.com/patrickqrim/CaltechFN/
[4] http://ufldl.stanford.edu/housenumbers/
[5] http://www.cs.toronto.edu/~kriz/cifar.html
[6] https://zenodo.org/records/7602025
[7] https://people.cs.umass.edu/~marlin/data.shtml
[8] https://www.kaggle.com/datasets/crawford/emnist
[9] https://www.kaggle.com/gpreda/chinese-mnist
[10] https://github.com/rois-codh/kmnist
[11] https://www.kaggle.com/c/Kannada-MNIST

2) [cDiffModel] Class-Conditional Diffusion Models

Similar to the [DiffModel] task, your task will be to implement a diffusion model. However, in this task, you will model the *class-conditional* distribution $p_\theta(\mathbf{x} \mid y)$, where y is the label of the image. To this end, use the labels in the dataset y and somehow pass this information into the U-Net, together with the noisy image. For example, you could create an embedding of y or use a one-hot vector to encode y. Again, clearly elaborate on the model selection process.

Analogously to [DiffModel], present your generated class-conditional data samples. Investigate the influence of several training hyperparameters and sampling hyperparameters. Demonstrate what happens if we use a condition vector that the network has not seen during training: For example, linearly interpolate between the encodings of y (one-hot or embeddings) and condition on this new vector.

3) [DiffModelInpaint] Diffusion Models for Image Inpainting

As in [DiffModel], train a diffusion model $p_\theta(\mathbf{x})$ on an image dataset. After training, we are interested in the task of *Image Inpaiting* (i.e., conditional sampling): Given a set of observed pixels $\mathbf{x}^{\text{obs}}$, we wish to draw a sample for the set of unobserved pixels $\mathbf{x}^{\text{unobs}} \sim p_\theta(\mathbf{x}^{\text{unobs}} \mid \mathbf{x}^{\text{obs}})$. For example, given the left half of an image, we wish to create a plausible continuation for the right half of the pixels.

Analogously to [DiffModel], first present your generated data samples (no inpainting). Then present the results of the inpainting task. Investigate the influence of several training hyperparameters and sampling hyperparameters.

4) [GAN] Generative Adversarial Networks

Your task will be to train a GAN consisting of a generator and discriminator architecture that is jointly optimized via the min-max training objective. Note that you will construct the model components based on a convolutional network architecture, and use two separate optimizers for gradient-based parameter updates that perform the objective minimization and maximization steps. At inference time, your GAN should be capable of synthesizing artificial data samples resembling to real examples.

Start by mathematically defining the min-max training objective and implementing the convolutional GAN which yields realistic data samples resembling the training dataset. Following a detailed model selection process, present your generated data samples. Investigate the learning progress balance and improvement tricks between the generator and discriminator networks.

For reference: Goodfellow et al., "Generative adversarial nets", NeurIPS 2014.

5) [cGAN] Conditional Generative Adversarial Networks

Your task will be to implement a variant of a traditional GAN consisting of a generator and discriminator, which now models a class-conditional data generation process $p_\theta(\mathbf{x} \mid y)$, where y is the label of the image. You will construct the model components based on a convolutional neural networks which also takes an

input class-condition vector into consideration, and use two separate optimizers for gradient-based parameter updates that implement the joint objective with minimization and maximization steps. At inference time, your cGAN should be capable of synthesizing class-specific artificial samples resembling to the real examples of a given class.

Start by mathematically defining the min-max training objective and implementing the class-conditional, convolutional GAN which yields realistic data samples. Following a detailed model selection process, present your generated class-conditional data samples. Investigate the learning progress balance and improvement tricks between the generator and discriminator networks.

For reference: M. Mirza and S. Osindero. "Conditional generative adversarial nets", arXiv preprint arXiv:1411.1784 (2014).

6) [**AutoEnc**] **Restoring Images with Autoencoders**

Your task will be to train an autoencoder that is capable of correcting and regenerating clean images from their distorted versions. To that end, you will construct an autoencoder by optimizing the parameters of a convolutional encoder-decoder pair that takes the distorted version of an image as the input, and the clean version of that image as the reconstructed output.

The trained autoencoder should be able to correct distortions on a test image such as: additive Gaussian noise on the input image, occlusion of a part of the input image with an arbitrary square patch, change in the brightness of the input image, rotations with a certain angle, horizontal/vertical flip of the input image, etc. Try to increase the generalization of the autoencoder to correct different distortions, and explore various latent bottleneck dimensionalities by assessing its relationship with model performance.

7) [β**-VAE**] **Beta Variational Autoencoders**

Your task will be to implement a variant of a variational autoencoder generative model using a convolutional encoder-decoder pair architecture. β-VAEs have a more generalized form than standard VAEs, with a similar training objective to the VAE loss function consisting of the reconstruction loss and Kullback-Leibler (KL) divergence loss terms, whereas the KL term in the loss function is weighted with a parameter β which controls the latent space regularization strength (for $\beta = 1$ it becomes a standard VAE).

Start by mathematically defining the training objective and implementing the model with convolutional encoder and decoder blocks which yields sufficient performance to generate artificial data samples. Following a detailed model selection process and demonstrating successfully generated samples, explore different aspects of your model such as: (i) How does varying the parameter β influence the training or generated samples? (ii) Explore and explain why the latent space of this model is enforced to have a more uncorrelated structure with independent components when β is very high?

For reference: Higgins et al., "β-VAE: Learning basic visual concepts with a constrained variational framework", ICLR 2017.

8) [cVAE] Conditional Variational Autoencoders

Your task will be to implement a variant of a variational autoencoder which models a class-conditional data generation process $p_\theta(\mathbf{x} \mid y)$. To this end, use the labels in the dataset y and somehow pass this information into the decoder neural network, together with the latent vector $\mathbf{z}$. For example, you could create an embedding of y or use a one-hot vector to encode y.

Start by mathematically defining the training loss function and implementing the model with convolutional encoder and decoder blocks which yields sufficient performance to generate class-conditional data samples. Following a detailed model selection process, present your generated class-conditional data samples. Demonstrate what happens if we pass a condition vector to the decoder that it has not seen during training: For example, linearly interpolate between the encodings of y (one-hot or dense embeddings) and condition on this vector.

For reference: Sohn et al., "Learning structured output representation using deep conditional generative models", NeurIPS 2015.

9) [RNN] Recurrent Neural Networks

This project does not relate to the image datasets that were previously discussed. It will address a miscellaneous audio or text dataset that *you will propose to us beforehand*, which will be considered for suitability.

Your task in this project will be to solve a customized audio or text classification problem using recurrent neural network (RNN) architectures. Start by creating an input encoder with embeddings to represent your sequential data in vector form. Construct and evaluate several RNN variants (e.g., a single output at the end of a sequence, or a bidirectional RNN) using either long short-term memory (LSTM) units or gated recurrent units (GRU) to accommodate for vanishing or exploding gradients. Investigate increasing the network depth by stacking several RNN layers. Discuss your architectural choices in model selection by comparing generalization performance.

10) [Own] Suggest Your Own Deep Learning Project

You are welcome to suggest your own deep learning project with a suitable workload. This broadly refers to any hands-on implementation and application of deep neural networks within the context of the course. Please contact us regarding your project topic until the set deadline and *propose your suggestion to us beforehand*, which will be considered for suitability. For inspiration, here is a list of interesting projects students have worked on in the past:

- Graph Neural Networks with Molecule Datasets
- Image Colorization with Diffusion Models (adding color to grayscale images)
- Generating Hyperspectral Image Data with cGANs
- Gesture Decoding from EEG Data using CNNs
- Music Genre Classification with RNNs

Submission details:

- In order to get a positive grade in this course, **you must receive at least 10 points on the project submission**.
- *Projects issued:* December 11th, 2024, 11:30
- *Deadline to choose a topic and group:* December 18th, 2024, 23:59
- *Deadline to submit projects:* January 22nd, 2025, 23:59
- *Submission:* Upload to TeachCenter a .zip file including a single PDF (report) and your code. Your code can consist of multiple `.py` and/or `.ipynb` files. Also include a `requirements.txt` file with the packages you have used. Clearly present your results in the report and make sure it is well structured.
- *Remarks:* In the PDF report, it is mandatory to use the cover sheet as the first page to indicate all group members. It is sufficient if only one of the group members submit the submission package on TeachCenter (code files and PDF report).