Deep Learning KU (708.220) WS21
Course Projects

Contact: Dr. Ozan Özdenizci (oezdenizci@tugraz.at)

This document consists of a list of projects that your group will choose one from. Overall goal of the course projects is to cover a variety of modern deep learning architectures and applications, distributed over a several groups. Note that any of the topics can be covered with applications to different datasets.

Overview: You will work on **only one of the models** from the *project topics* list below. This deep learning architecture will be **only applied to one dataset**. 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.

Project Reports: Submitted reports should include the following sections: (1) Introduction: describing the dataset and problem, (2) Methods: describing the neural network architecture with a mathematical definition of the loss function used for parameter optimization, (3) Results: demonstrating the model training and selection process with the performed comparisons, (4) Discussion: briefly describing your main observations. Include a clear description of your final deep learning architecture (e.g., regularization approaches, convolutional layer specifications, activations, latent dimensions, etc.). Report your model training details (e.g., loss function and optimization), and the amount of parameters in your networks. Provide tables that depicts your hyper-parameter search, as well as plots of training, validation and test set losses across training iterations.

Project Topics

1) **[AutoEnc] Correcting Images with Autoencoders**

   Your task will be to train an autoencoder that is capable of correcting and re-generating 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.

   Ideally the autoencoder should be able to correct distortions of various possibilities such as: additive Gaussian noise on the input image, occlusion of a part of the input image with a black 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.

This project can address a B&W or RGB colored image dataset of your choice (i.e., MNIST-like or CIFAR-10 like) as long as the computational demand is surmountable. There are several drop-in replacements of the standard MNIST dataset available, with images of handwritten digits/characters in different languages. You are encouraged to use one of these datasets, e.g., Chinese MNIST[1], Kuzushiji-MNIST[2] or Kannada-MNIST[3].

2) **[RNN] Recurrent Neural Networks for Text Classification**

Your task will be to solve a text classification problem based on input data represented as sequences, using recurrent neural network (RNN) architectures. Designed networks should have a recurrent structure in the hidden layers, and an output dense layer with the number of output units equal to the number of classes for text classification. This project, e.g., can address one of the following text datasets based on your preference: (i) Wikipedia Toxic Comments[4]: a large number of comments from Wikipedia labeled by human raters based on different toxic behavior categories, (ii) IMDB reviews[5]: a binary text classification task (for positive or negative sentiments) from a text database of movie reviews, (iii) Reuters newswire dataset[6]: a 46-class news topic classification task from text data of newswires.

Start by creating a text encoder with embeddings to represent your 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.

3) **[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 suf-

[1] `https://www.kaggle.com/gpreda/chinese-mnist`
[2] `https://github.com/rois-codh/kmnist`
[3] `https://www.kaggle.com/c/Kannada-MNIST`
[4] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`
[5] `https://keras.io/api/datasets/imdb/`
[6] `https://keras.io/api/datasets/reuters/`

ficient performance to generate artificial data samples. You are encouraged to use one of the previously discussed B&W image datasets (e.g., Chinese MNIST, Kuzushiji-MNIST or Kannada-MNIST). 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 further reference: I. Higgins et al., "β-VAE: Learning basic visual concepts with a constrained variational framework", ICLR 2017.

4) [cVAE] Conditional Variational Autoencoders

Your task will be to implement a variant of a variational autoencoder which models a conditional data generation process $p(x|c)$. In particular you will implement a cVAE that takes a one-hot encoded condition vector c as an additional input besides a sampled latent vector z to the decoder block. Since cVAEs do not perform classification as in discriminative models, you can exploit the given class labels y in the datasets as the condition vectors c such that the cVAE learns to generate class-specific data inputs.

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. You are encouraged to use any of the previously discussed B&W image datasets (e.g., Chinese MNIST, Kuzushiji-MNIST or Kannada-MNIST). Following a detailed model selection process, present your generated class-conditional data samples. Demonstrate how we can use cVAEs to synthesize novel images with an arbitrary condition c that possesses a particular input style.

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

5) [TransferL] Transfer Learning with Pre-Trained Models

Your task will be to solve a regular image classification problem using a convolutional neural network with a pre-trained model weight initialization scheme, and subsequent finetuning of weights. To illustrate, e.g., you can tackle a smaller-scale binary image classification task with a MobileNet architecture[7] using weight initialization with pre-trained model weights on a different task to reduce model training time. In this case you can use pre-trained model weights[8] provided by Tensorflow from a training process on the 1000-class image classification task on the ImageNet dataset.

[7] `https://keras.io/api/applications/mobilenet/`

[8] `https://www.tensorflow.org/tutorials/images/transfer_learning`

You will experimentally demonstrate the utility of knowledge transfer between different image classification tasks. Explore different approaches of finetuning, such as: (i) keeping the earlier convolutional feature extraction layers fixed and only finetuning the later layer weights, (ii) only finetuning the output classification layer, or (iii) finetuning all of the complete network weights, etc. Following a detailed description of the datasets involved in knowledge transfer and the used convolutional network architecture, investigate various finetuning pipelines (e.g., different optimizers and learning rate schedules, regularization schemes) to perform an appropriate optimization scheme for the new classification task. This project can address any smaller-scale image classification dataset (e.g., cats vs dogs[9], horses vs humans[10]) of your choice as long as the computational demand is surmountable.

Submission details:

- *Projects issued:* December 10th, 2021, 14:00
- *Deadline to choose a topic and group:* December 17th, 2021, 23:59
- *Deadline to submit projects:* January 18th, 2022, 23:59
- *Submission:* Upload to TeachCenter a .zip file including a single PDF (report) and your commented .py (code) files. Present your results clearly and structured in the report. Your code should be directly executable (assuming that the provided files are present at the working directory).
- *Remarks:* In the PDF report, indicate your group partners on top of the first page (write "Group partner: ⟨First name⟩, ⟨Last Name⟩, ⟨Matrikelnumber⟩". It is sufficient if only one of the group members submit the submission package on TeachCenter (codes and PDF report).

[9] `https://www.tensorflow.org/datasets/catalog/cats_vs_dogs/`
[10] `https://www.tensorflow.org/datasets/catalog/horses_or_humans/`