Final Project Options and Submission Guidelines

You can choose between two different types of final projects:

Research-based projects
These projects are centered around one or more scientific papers from the literature. Your work should include studying the paper, implementing (or re-implementing) the proposed method, analyzing the results, and preparing a brief discussion of the paper's contributions and limitations.

***Application-based project**s*
These projects do not refer to a specific research paper. Instead, they focus on applying neural networks to solve a practical task (e.g., image classification, sequence prediction, anomaly detection, generative modeling). The emphasis is on designing, training, and evaluating a model for a concrete problem.

Group work
Projects can be carried out in groups of up to **2–3** students. For each project topic, **a maximum of two groups** is allowed.

Project submission
For the submission, you only need to send me a presentation (slides) describing what you have done.
It is not necessary to send the code.
The submission deadline is **mid-month;** after that, **we will arrange a date for the oral discussion**.

Research-based projects
P1) *Paper: Goodfellow, Ian J., et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Implement a basic GAN (generator + discriminator) on a simple dataset (e.g., MNIST). Then explore how changes in architecture (number of layers, activation functions) or training dynamics (learning rate, batch size) impact mode collapse or sample quality.

P2) *Paper: Nwankpa, C. (2018). Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint arXiv:1811.03378.*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Run experiments comparing different activation functions (ReLU, GELU, Tanh, etc.) in a fixed architecture (e.g., a simple MLP or CNN) on a fixed dataset (e.g., MNIST). Evaluate training dynamics, convergence speed, and final test accuracy, then present which activations work best under which condition.

P3) *Paper: Apicella, Andrea, et al. "A survey on modern trainable activation functions." Neural Networks 138 (2021): 14-32.*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Implement one or more trainable activation functions (e.g., parametric ReLU) in a fixed architecture (e.g., a simple MLP or CNN) on a fixed dataset (e.g., MNIST), compare performance against fixed activations, and analyze results.

P4) *Paper: Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Re-implement the VAE described in the paper (encoder + decoder, reparameterization trick) on a simple dataset (e.g. MNIST). Train the model and visualize both reconstructions and samples from the latent space. Experiment by varying the latent dimension and network size, and analyze how these choices affect reconstruction quality and generative capability.

P5) *Paper: Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Implement a simple feedforward neural network from scratch (no high-level libraries) using the back-propagation algorithm. Train it on a small dataset (e.g., MNIST subset).
Experiment with different activation functions, hidden layer sizes, or learning rates, and analyze convergence behavior. PyTorch comparison: Implement the same network architecture using PyTorch and its autograd functionality. Compare training speed, ease of implementation, and numerical results (loss and accuracy) between the manual implementation and PyTorch.

P6) *Paper: LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper. Implement a minimal CNN from scratch, in particular implement a 2D convolutions (no built-in functions) with learnable filters, subsampling/pooling, nonlinear activation functions and fully connected output layer. Implement backpropagation through convolution (derive the gradient manually), Use only NumPy or similar low-level tools. Train it on a simple dataset (e.g MNIST or a subset of MNIST). Show internal representationsm, for example visualize learned convolution filters and intermediate feature maps. Compare the performance with a multilayer perceptron (MLP) of similar size.

P7) *Paper: "Empirical evaluation of the improved RProp learning algorithms, Christian Igel, Michael Husken, Neurocomputing, 2003"*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Following the paper, compare the classic resilient backpropagation (RProp) with at least two proposed variants of the algorithm as weight update methods (batch update) using a small dataset (e.g., MNIST). Fix the activation function and the number of internal nodes (at least three different dimensions), and compare the results obtained with the different learning algorithms.

P8) *Paper: Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. The journal of machine learning research, 13(1), 281-305*
Include a brief discussion of the main aspects of the paper, summarize the findings and critically analyze the methodology of the paper.
Use resilient backpropagation (RProp) as weight update algorithm, with a neural network having a single layer of internal nodes. Select the model hyperparameters, i.e., the RProp parameters ($\eta+$ and $\eta-$) and the number of internal nodes, based on a k-fold cross-validation approach (e.g., k= 10). Compare the classical "grid" approach with the "random" approach for hyperparameter search. Select and keep all other parameters constant, such as activation functions.

P9) *Paper: Prechelt, L. (2002). Early stopping-but when?. In Neural Networks: Tricks of the trade (pp. 55-69). Berlin, Heidelberg: Springer Berlin Heidelberg*
Provide a brief discussion of the paper, summarizing its main findings and critically evaluating the proposed methodology. Analyze the learning behavior of a neural network with a single hidden layer using sigmoid activations. Examine how different early-stopping criteria affect training performance by implementing the GL and PQ algorithms with multiple values of the parameter α. Test networks with at least five different hidden-layer sizes, use a fixed update rule (e.g., resilient backpropagation, Rprop), and run experiments on a simple dataset such as MNIST.

Application-based projects

P10) Use the raw MNIST images as input for a classification task with C=10 classes. Construct a dataset of N input–label pairs and split it into training and test sets (use at least 10,000 samples for training and 2,500 for testing). Adopt resilient backpropagation (RProp) as the weight update rule (batch mode).
Build k autoencoders, each with a single hidden layer of size m_h, where $h=1,2,\ldots,k$ and $k=5$ (for example $m_1=25$, $m_2=50$, $m_3=75$, $m_4=100$, $m_5=125$). This yields k encoders E_h, which project each input x into a different representation. For each encoder E_h, study the learning dynamics of a neural network with a single hidden layer and sigmoid activation functions, taking $E_h(x)$ as input. Examine metrics such as the number of epochs required for convergence and the classification accuracy on the test set.

P11) Use the raw MNIST images as input for a 10-class classification task. Build a dataset of N input–label pairs and split it into training and test sets (at least 10,000 training samples and 2,500 test samples).
Train the models using gradient descent with momentum.
Study the learning behavior of a neural network with a single hidden layer by varying the learning rate η, the momentum coefficient, and the number of hidden units (use at least five different hidden-layer sizes). Keep all other architectural choices fixed, including the output activation functions. For each configuration, analyze the training process (epochs to convergence, trends in training and validation errors, test accuracy). Additionally, compare the stability of training under different hyperparameter settings, examine the sensitivity to initialization (run each experiment at least twice with different random seeds), and report any systematic patterns you observe.

P12) Use the raw MNIST images as input for a 10-class classification task. Construct a dataset of N input–label pairs and split it into training and test sets (at least 10,000 samples for training and 2,500 for testing).
Train the models using standard gradient descent.
Study the learning behavior of a neural network with a single hidden layer by varying the training modality: online, batch, and mini-batch learning. Perform this analysis for at least three different hidden-layer sizes while keeping the activation functions fixed. For each configuration, evaluate the learning dynamics (epochs to convergence, training and validation error curves, test accuracy). Additionally, examine the impact of the mini-batch size.

P13) Use the raw MNIST images as input for a 10-class classification task. Construct a dataset of N input–label pairs and split it into training and test sets (at least 10,000 for training and 2,500 for testing).
Train all models using resilient backpropagation (RProp) in batch mode.
Study the learning behavior of a neural network by varying the number of hidden layers from 1 to 5. For each depth, test multiple choices for the number of hidden units and compare three activation

functions: tanh, ReLU, and leaky ReLU. For each configuration, analyze the training dynamics—epochs to convergence, trends in training and validation errors, and final test accuracy. Additionally, repeat selected experiments with different initializations to confirm the robustness of your findings.

P14) Use the raw MNIST images as input for a 10-class classification task. Build a dataset of N input–label pairs and split it into training and test sets (at least 10,000 samples for training and 2,500 for testing). Use the same weight update algorithm—either resilient backpropagation (RProp) or Adam—for all experiments.
Compare the performance of a fully connected neural network (FCNN) and a convolutional neural network (CNN) under systematic variation of their respective hyperparameters. For the FCNN, explore different numbers of hidden layers, different hidden-layer sizes, and activation functions. For the CNN, vary the number of convolutional filters, kernel sizes, and the number of convolution–pooling blocks.
For each architecture and hyperparameter configuration, analyze the learning dynamics and final performance (epochs to convergence, training and validation error curves, test accuracy).

NOTE THAT: For all projects, unless otherwise specified, you may either use a standard neural network library such as PyTorch or develop your own neural network implementation from scratch