Generative Computer Vision: Challenge

Handout: 12.05.2025 08:00
Due: 20.05.2025 23:59

Spring 2025
Hands-On Deep Learning

Challenge Open in IDE

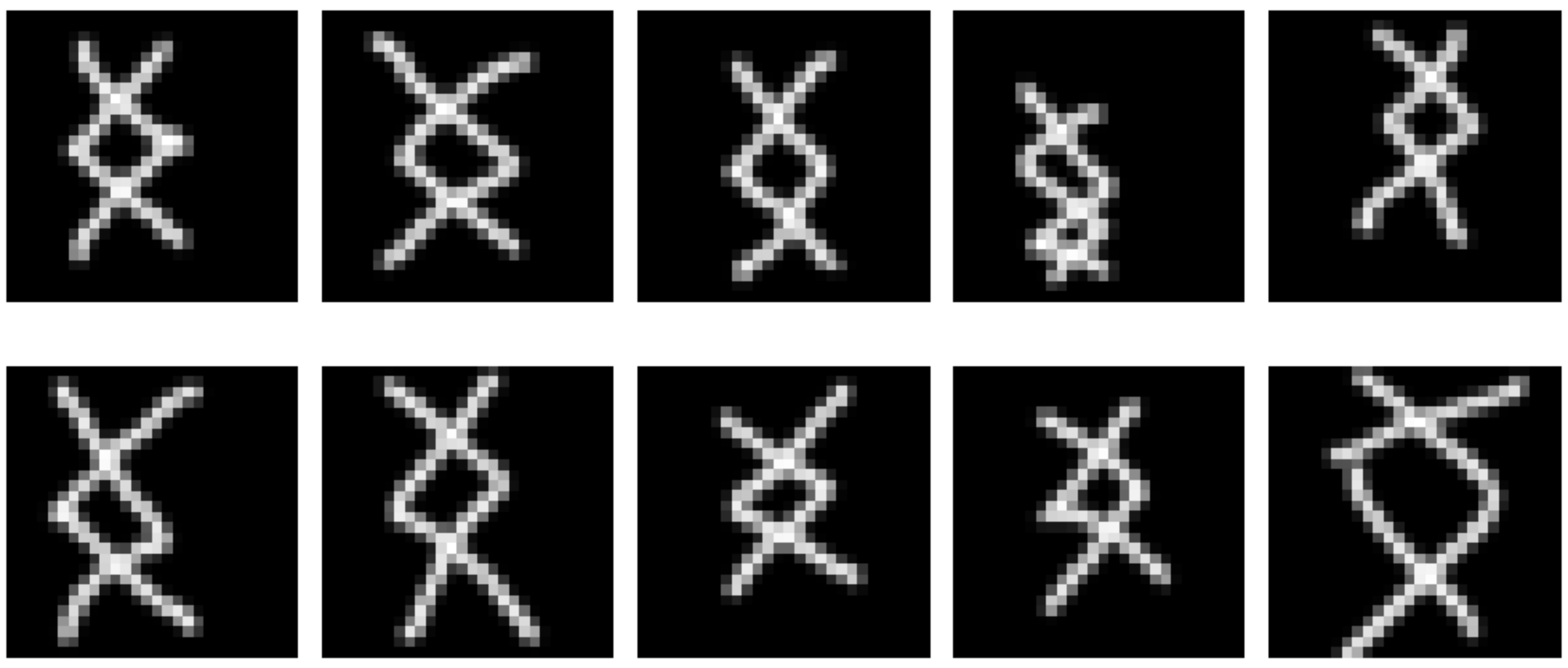
Learning an alphabet from 10 examples.

Prototype Notebook: Open in Google Colab.

Important: you have to submit your solution on CodeExpert.

The Omniglot dataset was released in 2015 as a benchmark for sample efficient learning. It contains 1623 classes of handwritten characters each containing only ≈ 20 examples.

Your challenge: you will be given an MNIST rectified flow model trained for 100 epochs using the code in the *Diffusion models* and *Conditional image generation* sections of the notebook. You will need to adapt this model to generate new examples of an Anglo-Saxon rune from Omniglot, using only 10 samples in your train dataset.



Spec

Your task is to fill out the `finetune_model` function, which will be called separately for each unet and rune. This function will be passed unets with the following hyperparameters: `ConditionalDiffusionUNet(base_dim=32, n_classes=10, n_updown_blocks=2, n_middle_blocks=8)`, as well as a 10 example train dataset for a rune that you need to finetune your model on. The runes will be assigned the new label `c=10` at training & evaluation time.

The `finetune_model` function will be called the following way:

```python
def run():
  scores = []

  train_datasets, test_datasets = get_data()

  for i in range(1, 6):

    set_seed(42 + i)

    # Get datasets for training and testing
    train_dataset = train_datasets[i-1]
    test_dataset = test_datasets[i-1]

    # Load pretrained model
    pretrained_model = load_pretrained_model(i)

    # Train the model using student's train_model function
    finetuned_model = finetune_model(pretrained_model, train_dataset)

    # Evaluate the model on the test set
    score = evaluate_model(finetuned_model, test_dataset)
    print(f"Score for model {i} rune {i}: {score}")
    scores.append(score)

  scores = np.array(scores)
  mean_score, std_score = np.mean(scores), np.std(scores)
  print(f"Mean score: {mean_score:.6f}")
  score = mean_score

  return score
```

The following can be used to draw many samples of t to validate a final model:

```python
def validate_rectified_flow_conditional_multit(model, dataloader, t_samples):
    # Uses multiple samples of t to get a better estimate of the loss
    losses = [
        validate_rectified_flow_conditional(model, dataloader) for _ in tqdm.trange(t_samples)
    ]
    losses = torch.tensor(losses)
    return losses.mean(), losses.std()
```

Scoring System

Points are awarded based on the mse score for the rectified flow objective, according to the thresholds in the table below.

Your solution will be evaluated using 5 runs with different, held-out runes and pretrained model weights, using 1000 samples of t. The number of samples in the train dataset and the model architecture will be kept constant.

Some runes are consistently harder to learn than others, the rune generated in the dataset above is a small amount harder than the average test rune.

Rectified flow loss (running)	CodeExpert score (submitting)	Points Earned
> 0.150	< 8.3%	0
≤ 0.150	≥ 8.3%	1
≤ 0.120	≥ 25.0%	2
≤ 0.100	≥ 41.7%	3
≤ 0.094	≥ 58.3%	4
≤ 0.088	≥ 75.0%	5
≤ 0.083	≥ 91.7%	6

We recommend inspecting generated images from your model as part of your solution development process, this can be done in the prototyping notebook. The rectified flow objective on its own can be quite opaque and noisy without a large number of t samples,

Rules

- No outside data or models are allowed.
- You can use the installed python packages and code from the notebook provided you don't use them to break the first rule.
- One call of `finetune_model` in your solution should take less than 2.5 minutes. We cannot guarantee that the time taken per evaluation on the cluster will be the same as whatever environment your use to develop you solution, so we recommend leaving a margin of error here.

Tutorial

Tutorial on how to use CodeExpert.

Toblerone

The top three solutions will receive a Toblerone. Good luck! 🚀