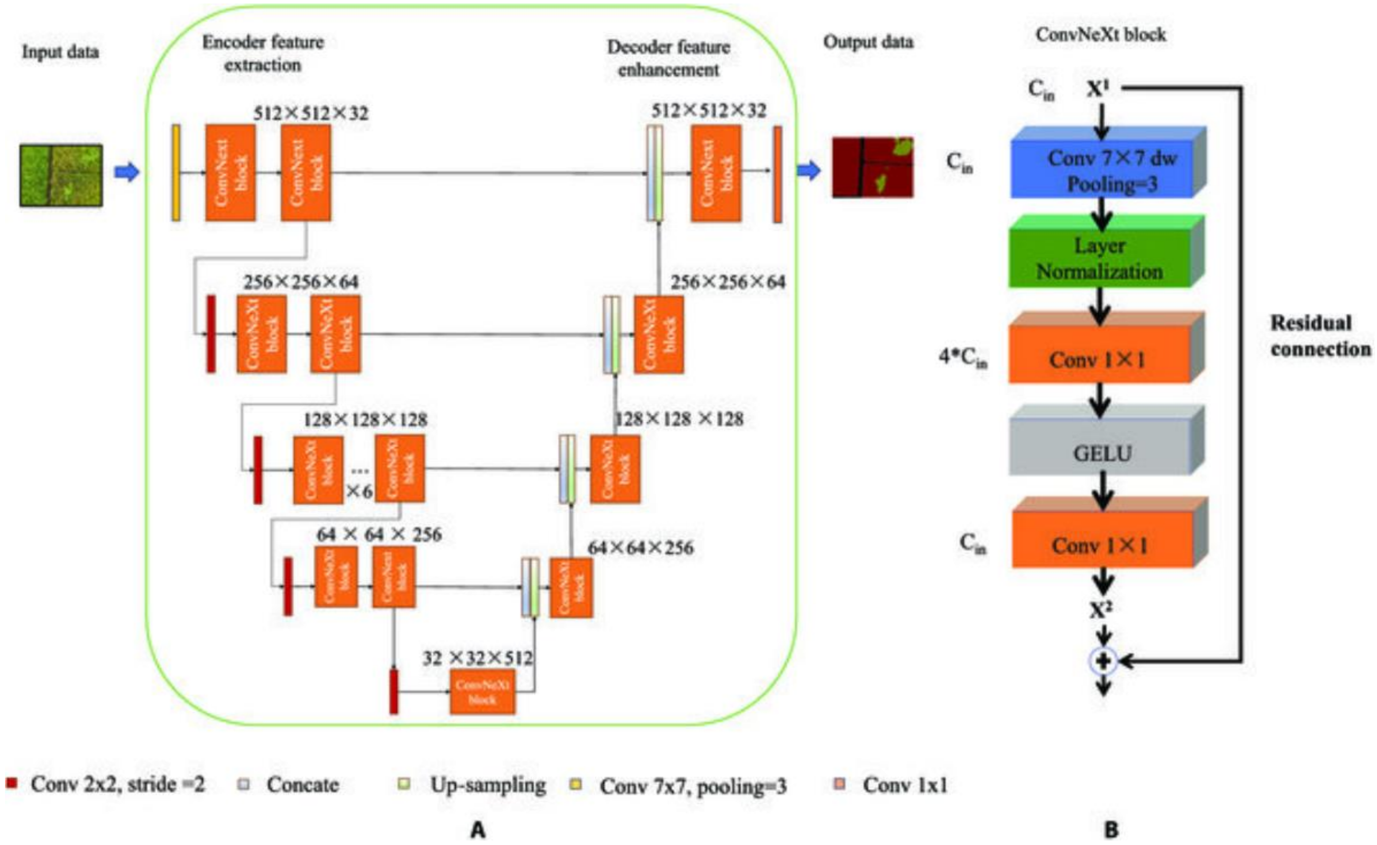
Predicting Emergent Dynamics

Project 01

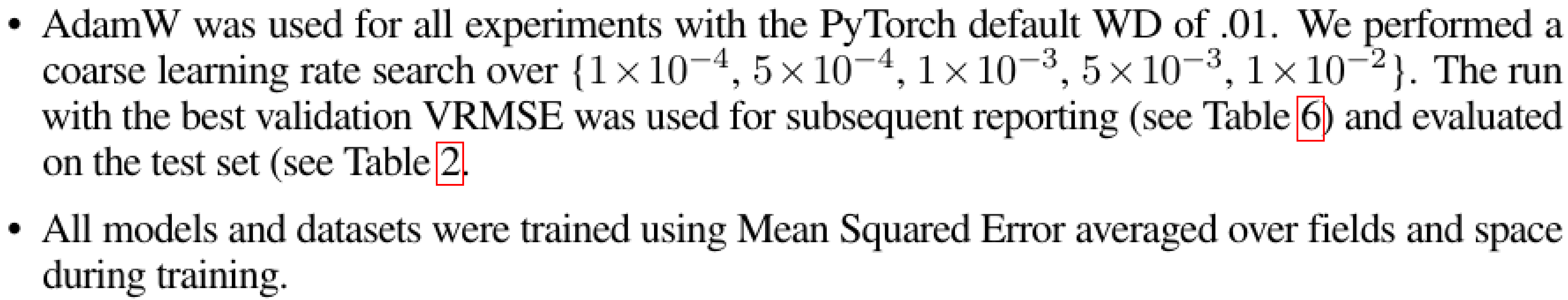
UNetConvNext:

Implementation of the U-Net model using ConvNext blocks.

- CNextU-net
  - Spatial filter size - 7
  - Initial dimension - 42
  - Blocks per stage - 2
  - Up/Down blocks - 4
  - Bottleneck blocks - 1

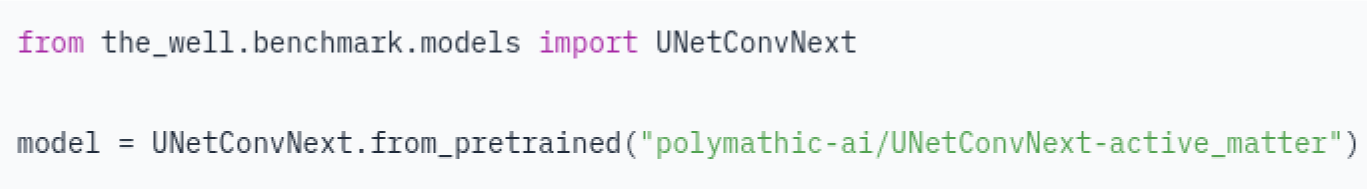


UNetConvNext:



- AdamW was used for all experiments with the PyTorch default WD of .01. We performed a coarse learning rate search over $\{1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}, 1\times10^{-2}\}$. The run with the best validation VRMSE was used for subsequent reporting (see Table 6) and evaluated on the test set (see Table 2.
- All models and datasets were trained using Mean Squared Error averaged over fields and space during training.

https://github.com/PolymathicAI/the_well/blob/master/the_well/benchmark/models/unet_convnext/__init__.py



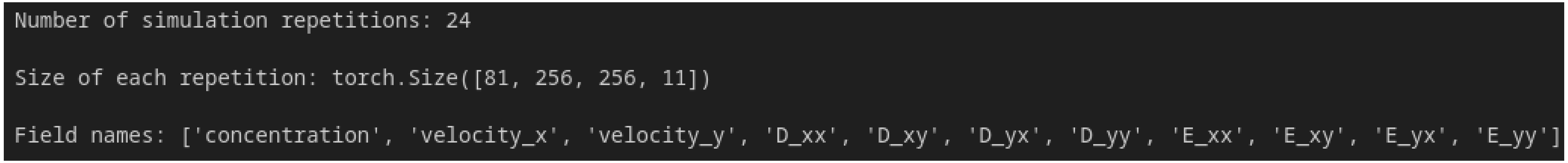
```
from the_well.benchmark.models import UNetConvNext

model = UNetConvNext.from_pretrained("polymathic-ai/UNetConvNext-active_matter")
```

UNetConvNext:

Input – Output:

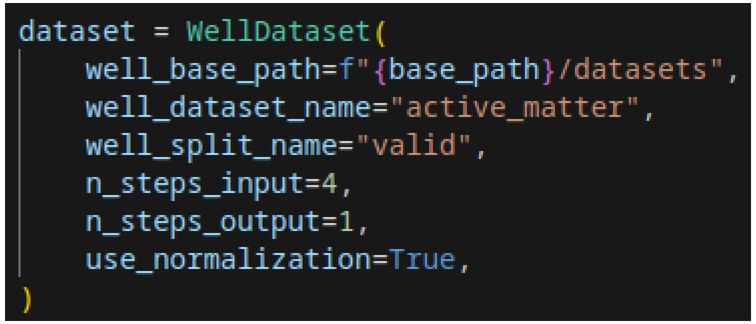
The baselines are trained on the forward problem - predicting the next snapshot of a given simulation from a short history of 4 time-steps.



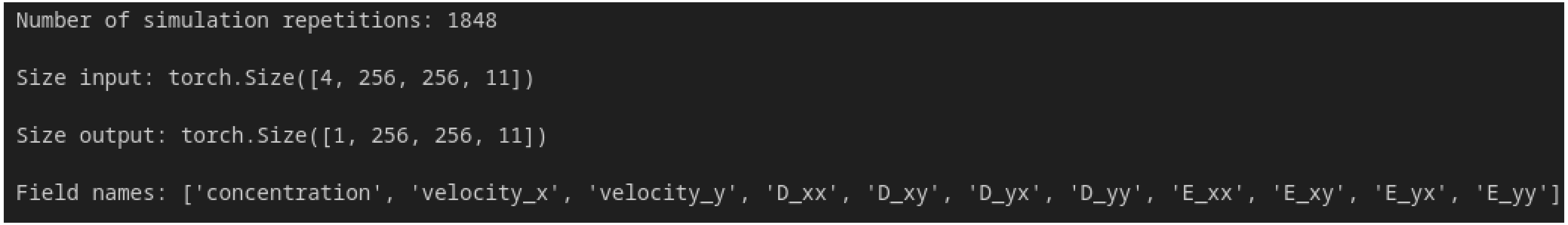
```
Number of simulation repetitions: 24

Size of each repetition: torch.Size([81, 256, 256, 11])

Field names: ['concentration', 'velocity_x', 'velocity_y', 'D_xx', 'D_xy', 'D_yx', 'D_yy', 'E_xx', 'E_xy', 'E_yx', 'E_yy']
```



```
dataset = WellDataset(
    well_base_path=f"{base_path}/datasets",
    well_dataset_name="active_matter",
    well_split_name="valid",
    n_steps_input=4,
    n_steps_output=1,
    use_normalization=True,
)
```



```
Number of simulation repetitions: 1848

Size input: torch.Size([4, 256, 256, 11])

Size output: torch.Size([1, 256, 256, 11])

Field names: ['concentration', 'velocity_x', 'velocity_y', 'D_xx', 'D_xy', 'D_yx', 'D_yy', 'E_xx', 'E_xy', 'E_yx', 'E_yy']
```

Pseudocode:

```
1. Import libraries
2. Load the data for the forward problem (Input=4, Output=1)
3. Import/initialize the model
4. Train with the respective opt and error
5. Test with the validation set
```