Homework #2, 繳交期限 2024/10/25
Hagen-Poiseuille flow

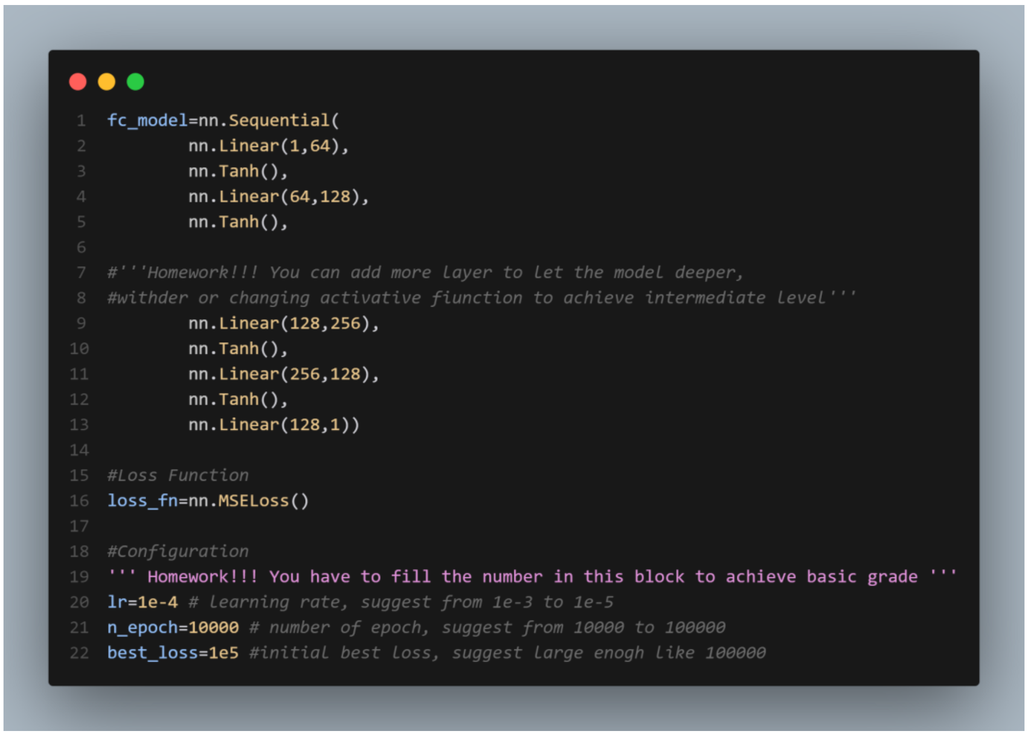
系所：工海所

姓名：劉梓

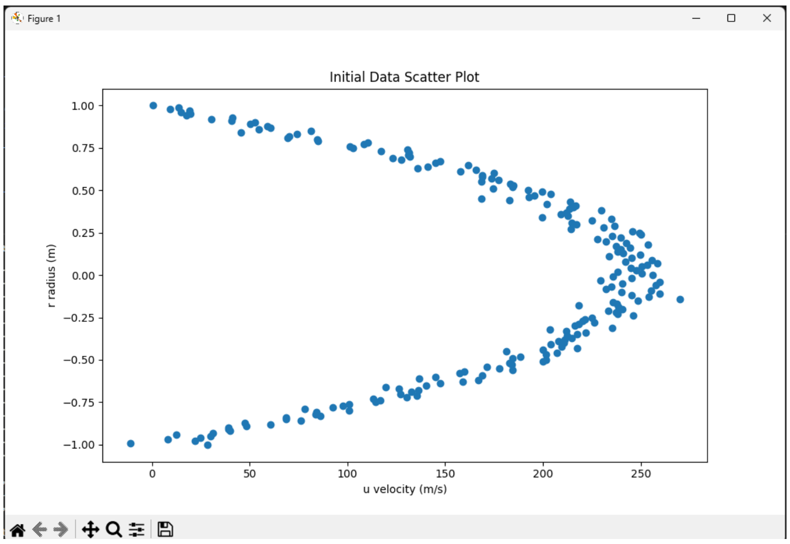
學號：R13525030

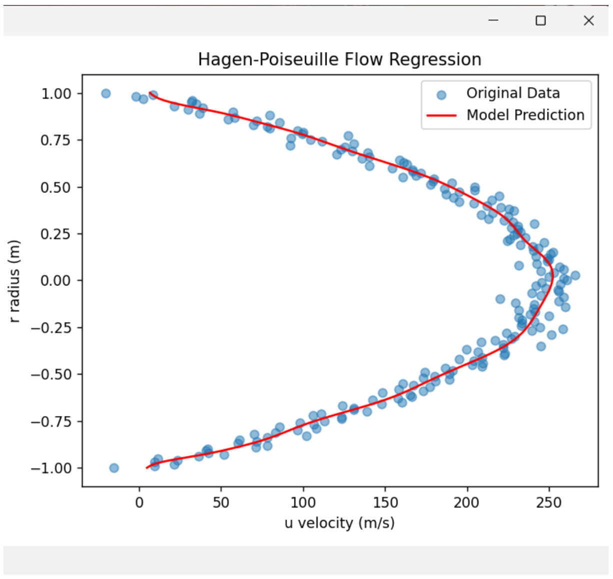
Basic

Code：



```python
fc_model=nn.Sequential(
        nn.Linear(1,64),
        nn.Tanh(),
        nn.Linear(64,128),
        nn.Tanh(),

#'''Homework!!! You can add more layer to let the model deeper,
#withder or changing activative fiunction to achieve intermediate level'''
        nn.Linear(128,256),
        nn.Tanh(),
        nn.Linear(256,128),
        nn.Tanh(),
        nn.Linear(128,1))

#Loss Function
loss_fn=nn.MSELoss()

#Configuration
''' Homework!!! You have to fill the number in this block to achieve basic grade '''
lr=1e-4 # learning rate, suggest from 1e-3 to 1e-5
n_epoch=10000 # number of epoch, suggest from 10000 to 100000
best_loss=1e5 #initial best loss, suggest large enogh like 100000
```





Homework #2, 繳交期限 2024/10/25
Hagen-Poiseuille flow

系所：工海所

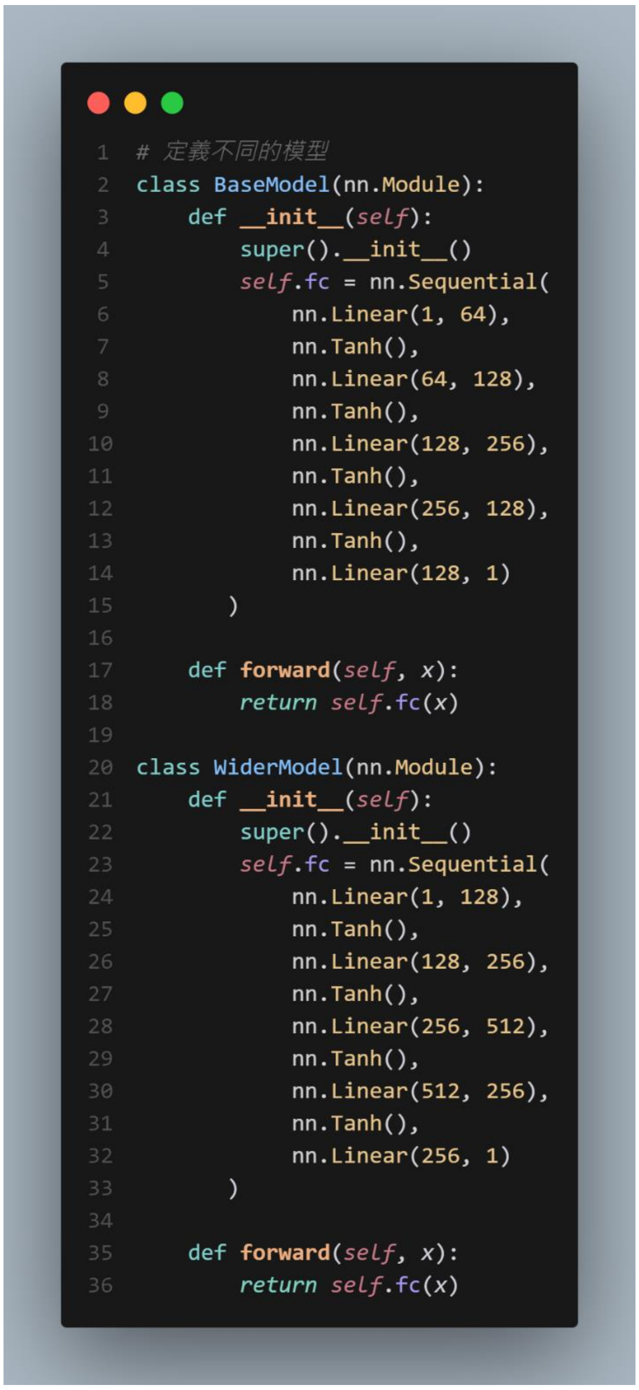
姓名：劉樺

學號：R13525030

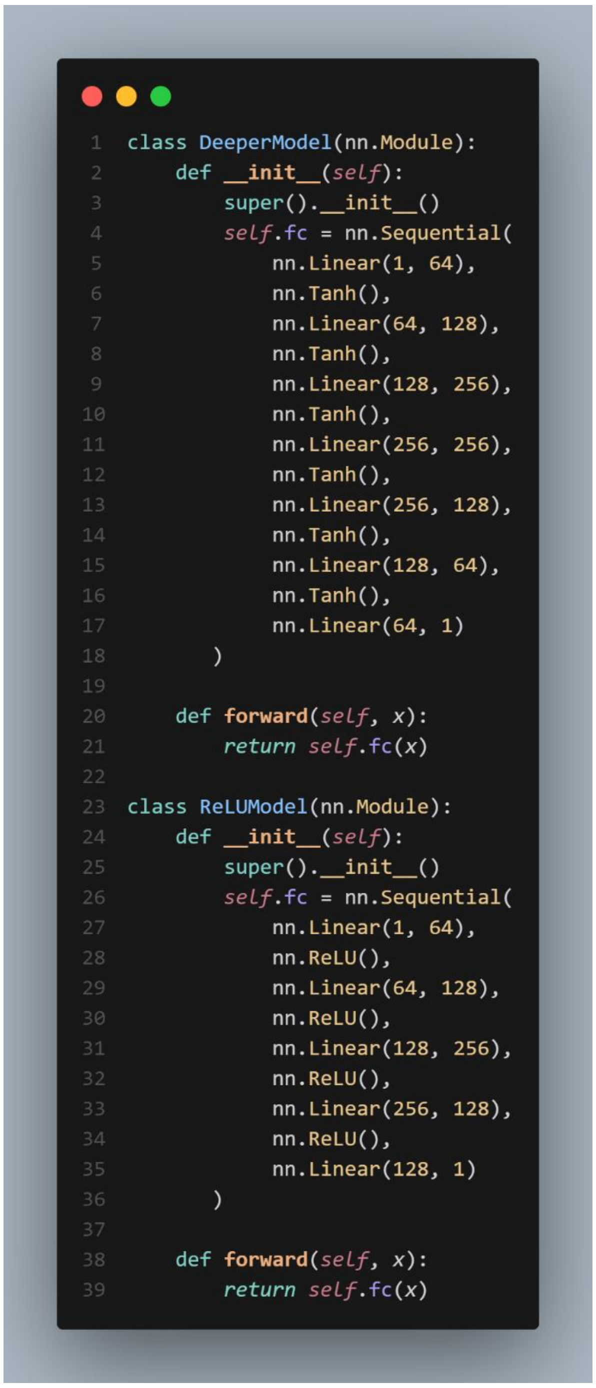
Intermediate

(1) 可嘗試更改模型(加寬、加深、激活函數.....)以提升擬合能力。
(2) 分別說明你所觀察到加寬模型、加深模型與更改激活函數對loss造成之影響，並嘗試說明可能原因。

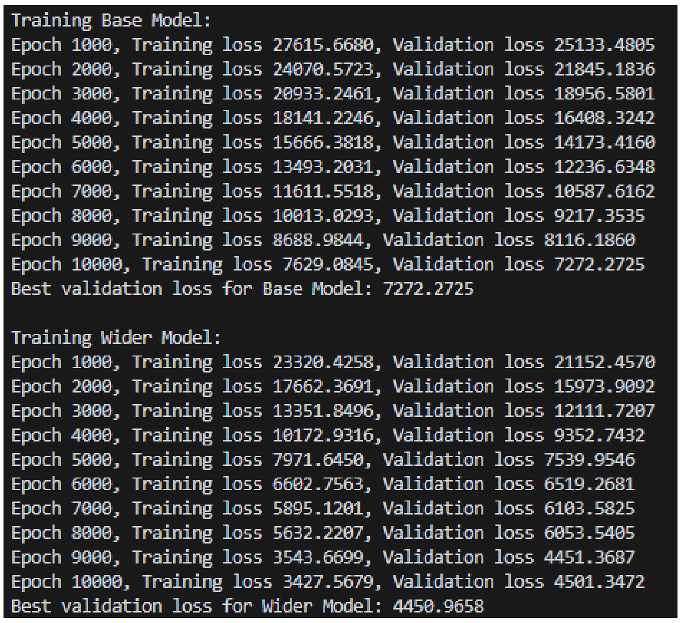
Code：



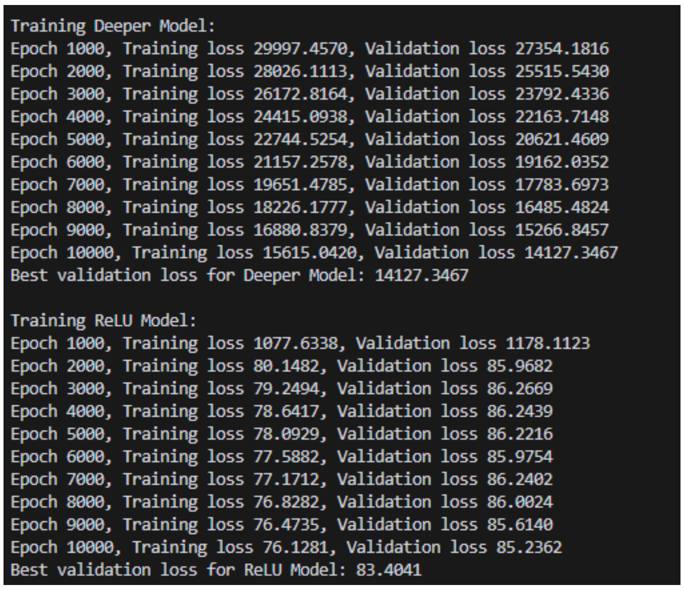
```python
# 定義不同的模型
class BaseModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(1, 64),
            nn.Tanh(),
            nn.Linear(64, 128),
            nn.Tanh(),
            nn.Linear(128, 256),
            nn.Tanh(),
            nn.Linear(256, 128),
            nn.Tanh(),
            nn.Linear(128, 1)
        )

    def forward(self, x):
        return self.fc(x)

class WiderModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(1, 128),
            nn.Tanh(),
            nn.Linear(128, 256),
            nn.Tanh(),
            nn.Linear(256, 512),
            nn.Tanh(),
            nn.Linear(512, 256),
            nn.Tanh(),
            nn.Linear(256, 1)
        )

    def forward(self, x):
        return self.fc(x)
```



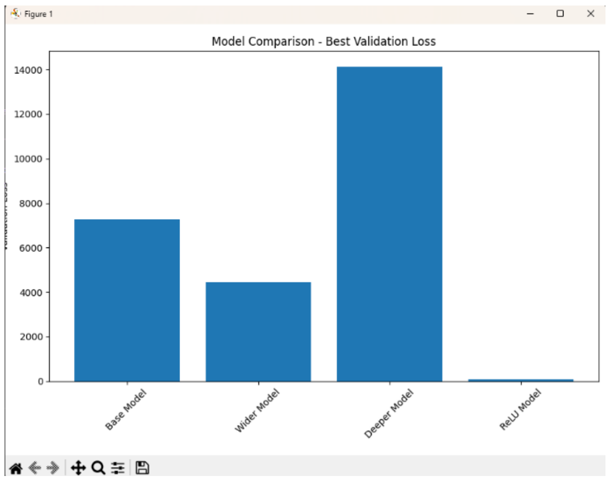
```python
class DeeperModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(1, 64),
            nn.Tanh(),
            nn.Linear(64, 128),
            nn.Tanh(),
            nn.Linear(128, 256),
            nn.Tanh(),
            nn.Linear(256, 256),
            nn.Tanh(),
            nn.Linear(256, 128),
            nn.Tanh(),
            nn.Linear(128, 64),
            nn.Tanh(),
            nn.Linear(64, 1)
        )

    def forward(self, x):
        return self.fc(x)

class ReLUModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(1, 64),
            nn.ReLU(),
            nn.Linear(64, 128),
            nn.ReLU(),
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, 1)
        )

    def forward(self, x):
        return self.fc(x)
```

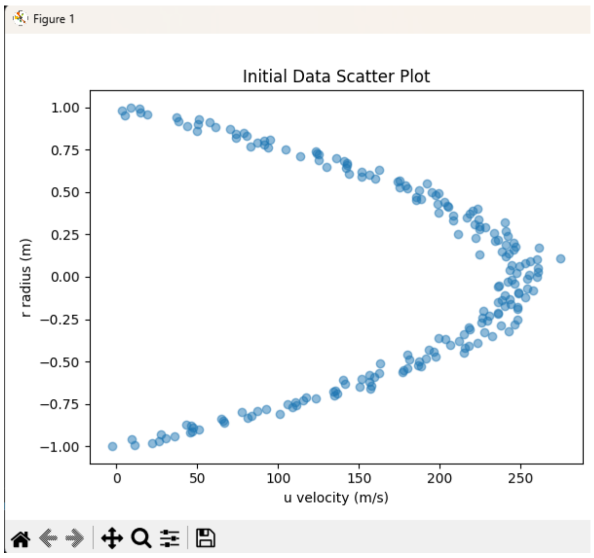


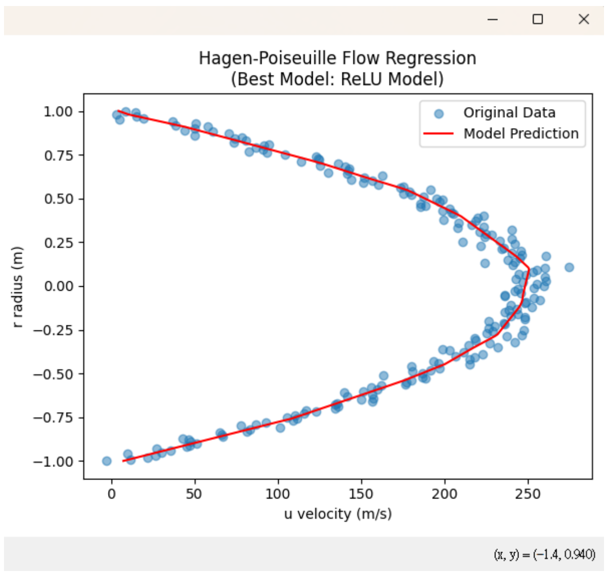
```
Training Base Model:
Epoch 1000, Training loss 27615.6680, Validation loss 25133.4805
Epoch 2000, Training loss 24070.5723, Validation loss 21845.1836
Epoch 3000, Training loss 20933.2461, Validation loss 18956.5801
Epoch 4000, Training loss 18141.2246, Validation loss 16408.3242
Epoch 5000, Training loss 15666.3818, Validation loss 14173.4160
Epoch 6000, Training loss 13493.2031, Validation loss 12236.6348
Epoch 7000, Training loss 11611.5518, Validation loss 10587.6162
Epoch 8000, Training loss 10013.0293, Validation loss 9217.3535
Epoch 9000, Training loss 8688.9844, Validation loss 8116.1860
Epoch 10000, Training loss 7629.0845, Validation loss 7272.2725
Best validation loss for Base Model: 7272.2725

Training Wider Model:
Epoch 1000, Training loss 23320.4258, Validation loss 21152.4570
Epoch 2000, Training loss 17662.3691, Validation loss 15973.9092
Epoch 3000, Training loss 13351.8496, Validation loss 12111.7207
Epoch 4000, Training loss 10172.9316, Validation loss 9352.7432
Epoch 5000, Training loss 7971.6450, Validation loss 7539.9546
Epoch 6000, Training loss 6602.7563, Validation loss 6519.2681
Epoch 7000, Training loss 5895.1201, Validation loss 6103.5825
Epoch 8000, Training loss 5632.2207, Validation loss 6053.5405
Epoch 9000, Training loss 3543.6699, Validation loss 4451.3687
Epoch 10000, Training loss 3427.5679, Validation loss 4501.3472
Best validation loss for Wider Model: 4450.9658
```



```
Training Deeper Model:
Epoch 1000, Training loss 29997.4570, Validation loss 27354.1816
Epoch 2000, Training loss 28026.1113, Validation loss 25515.5430
Epoch 3000, Training loss 26172.8164, Validation loss 23792.4336
Epoch 4000, Training loss 24415.0938, Validation loss 22163.7148
Epoch 5000, Training loss 22744.5254, Validation loss 20621.4609
Epoch 6000, Training loss 21157.2578, Validation loss 19162.0352
Epoch 7000, Training loss 19651.4785, Validation loss 17783.6973
Epoch 8000, Training loss 18226.1777, Validation loss 16485.4824
Epoch 9000, Training loss 16880.8379, Validation loss 15266.8457
Epoch 10000, Training loss 15615.0420, Validation loss 14127.3467
Best validation loss for Deeper Model: 14127.3467

Training ReLU Model:
Epoch 1000, Training loss 1077.6338, Validation loss 1178.1123
Epoch 2000, Training loss 80.1482, Validation loss 85.9682
Epoch 3000, Training loss 79.2494, Validation loss 86.2669
Epoch 4000, Training loss 78.6417, Validation loss 86.2439
Epoch 5000, Training loss 78.0929, Validation loss 86.2216
Epoch 6000, Training loss 77.5882, Validation loss 85.9754
Epoch 7000, Training loss 77.1712, Validation loss 86.2402
Epoch 8000, Training loss 76.8282, Validation loss 86.0024
Epoch 9000, Training loss 76.4735, Validation loss 85.6140
Epoch 10000, Training loss 76.1281, Validation loss 85.2362
Best validation loss for ReLU Model: 83.4041
```







加寬模型：

- 影響：通常會降低 loss，因為模型有更多的參數來擬合數據
- 原因：更寬的層能夠學習更多的特徵，增加了模型的表達能力

加深模型：

- 影響：可能會降低 loss，但也可能導致訓練困難
- 原因：更深的網絡可以學習更複雜的函數，但也可能面臨梯度消失/爆炸問題

更改激活函數（使用 ReLU）：

- 影響：可能會改善訓練速度和最終性能
- 原因：ReLU 可以緩解梯度消失問題，允許更深的網絡更容易訓練。但它也可能導致"死亡 ReLU"問題

Homework #2, 繳交期限 2024/10/25
Hagen-Poiseuille flow

系所：工海所

姓名：劉梓

學號：R13525030

Advanced

為避免overfitting，可使用.ckpt檔將模型之參數儲存起來，並且於預測時讀取模型參數。比較該epoch與上一個epoch之validation loss，若該epoch之validation loss較小即可儲存該模型參數，並於最後繪圖時讀取並繪圖。嘗試使用torch內建之方法儲存與讀取模型參數，並於最後繪圖時讀取並繪圖。(15%)

Code：

在 train_model 函數中，在每次找到更好的驗證損失時都會儲存模型參數：

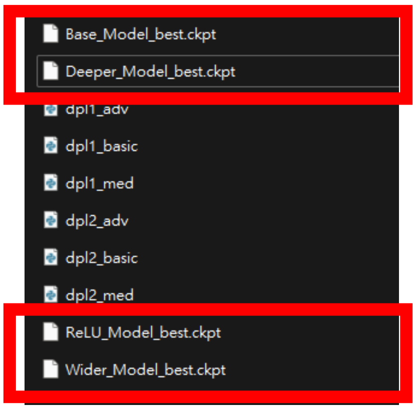
```
if val_loss < best_loss:
    best_loss = val_loss.item()
    torch.save(model.state_dict(), f"{model_name}_best.ckpt")
```

在訓練循環中，為每個模型指定了一個唯一的名稱，用於儲存檢查點文件：

```
best_loss = train_model(model, train_set_r, train_set_u, val_set_r, val_set_u,
model_name=name.replace(" ", "_"))
```

在繪製最終預測圖之前，加載了最佳模型的參數：

```
best_model.load_state_dict(torch.load(f"{best_model_name.replace(' ',
'_')}_best.ckpt"))
```


Base_Model_best.ckpt
Deeper_Model_best.ckpt
dpl1_adv
dpl1_basic
dpl1_med
dpl2_adv
dpl2_basic
dpl2_med
ReLU_Model_best.ckpt
Wider_Model_best.ckpt

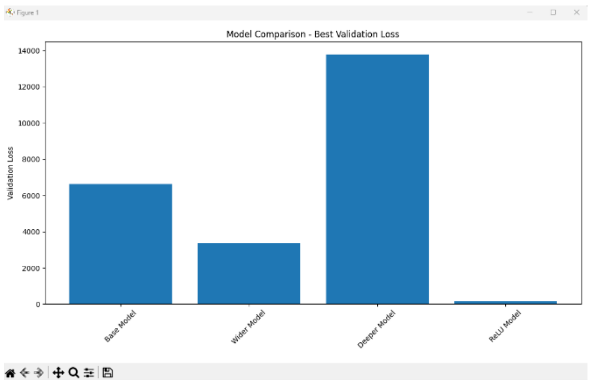


Figure 1
Model Comparison - Best Validation Loss
Validation Loss
Base Model
Wider Model
Deeper Model
ReLU Model

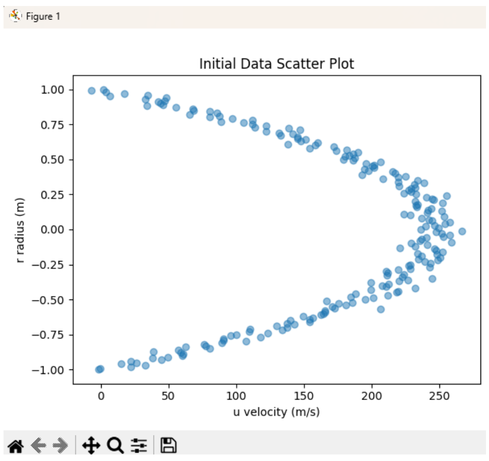
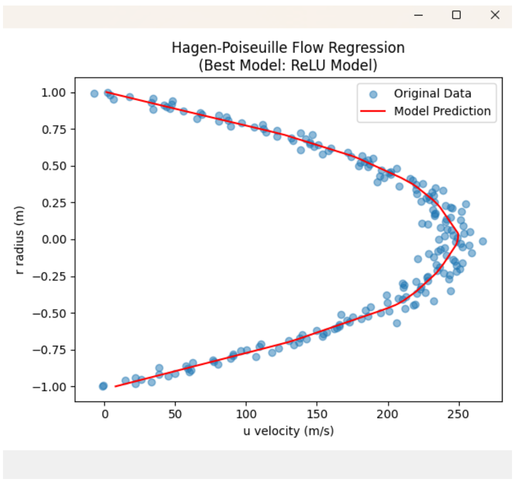


Figure 1
Initial Data Scatter Plot
r radius (m)
u velocity (m/s)


Hagen-Poiseuille Flow Regression
(Best Model: ReLU Model)
Original Data
Model Prediction
r radius (m)
u velocity (m/s)