

Documentation pytorch du LSTM : <https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html?highlight=lstm#torch.nn.LSTM>

Ici pour notre modèle nous choisissons une taille de 150 neurones pour la couche cachée. Avec une seule couche récurrente (une cellule LSTM). Input_size correspond au nombre de features qui seront passés dans le vecteur x. Ici nous avons qu'une seule feature à savoir la valeur à la clôture. Ne pas confondre input_size avec sequence length. Le premier correspond au nombre de features (caractéristiques) tandis que le second correspond à la taille du vecteur x qui est passé en entrée.

```
In [ ]: class LSTM(nn.Module):
    def __init__(self, input_size=1, hidden_layer_size=150, output_size=1, num_layers=1):
        super().__init__()
        self.hidden_layer_size = hidden_layer_size
        self.lstm = nn.LSTM(input_size, hidden_layer_size, num_layers, batch_first = True)
        # linear layer pour prédire
        self.linear = nn.Linear(hidden_layer_size, output_size)

    def forward(self, input_batch):
        self.hidden_cell = (torch.zeros(1, input_batch.size(0), self.hidden_layer_size, device=self.device),
                             torch.zeros(1, input_batch.size(0), self.hidden_layer_size, device=self.device))
        lstm_out, self.hidden_cell = self.lstm(input_batch, self.hidden_cell)
        predictions = self.linear(lstm_out[:, -1, :])
        return predictions
```

On définit maintenant le modèle skorch pour pouvoir faire notre entraînement.

On réalise un GridSearch pour déterminer le meilleur nombre de neurones. On réalise les calculs sur la carte graphique.

Voici la documentation de Skorch pour plus d'information :

<https://skorch.readthedocs.io/en/stable/index.html>

```
In [ ]: net_regr = NeuralNetRegressor(
    LSTM,
    optimizer=torch.optim.Adam,
    max_epochs=1000,
    lr=0.001,
    callbacks=[
        LRScheduler(policy='StepLR', step_size=100, gamma=0.1)
    ],
    criterion=nn.MSELoss,
    device='cuda'
)
torch.cuda.empty_cache()
#paramètre qu'on va déterminer avec le GridSearch
param_grid = {'module__hidden_layer_size': [25, 50, 75, 100, 125, 150, 175, 200]}

grid = GridSearchCV(net_regr, param_grid, cv=3, error_score='raise')

#on réalise l'entraînement
grid.fit(train_features, train_labels)

# Sauvegarde du modèle
saveModel = datetime.datetime.now().strftime('%Y%m%d_%H_%M')
# cf doc for more information https://skorch.readthedocs.io/en/stable/user/save_load.html
joblib.dump(grid.best_estimator_, 'saving_model/model/'+saveModel+'.pkl')

# loading need to initialize net
# net_regr.initialize() # This is important!
```

```
# net_regr.load_params(f_params='some-file.pkl')

# on affiche le meilleur modèle et ses paramètres
print(grid.best_params_)
print(grid.best_estimator_)
```