Meta-learning Symmetries by Reparameterization for Rotated MNIST

JULIEN BERTRAND

BAPTISTE COEFFIER

Sommaire

- Papier de recherche
- Adaptation Rotated MNIST
- Modèle à circuit unique
- Modèle à circuit dédié
- Visualisation des angles et du weight sharing
- Résultats
- Conclusion

Meta-Learning Symmetries by Reparameterization‘

- Apprentissage automatique des symétries
- Pourquoi la reparamétrisassions ?
- Pourquoi cette architecture ?

Adaptation Rotated MNIST

```python
def load_rmnist_task_data(loader, num_tasks, k_spt, k_qry):
    """Transformation des données pour créer les ensembles support et requête"""
    task_data = []
    for batch_idx, (images, labels, angles) in enumerate(loader):
        x_spt, y_spt, angles_spt = images[:k_spt],
                                   labels[:k_spt],
                                   angles[:k_spt]
        x_qry, y_qry, angles_qry = images[k_spt:k_spt + k_qry],
                                   labels[k_spt:k_spt + k_qry],
                                   angles[k_spt:k_spt + k_qry]

        task_data.append((x_spt, y_spt, angles_spt, x_qry, y_qry, angles_qry))
        if len(task_data) >= num_tasks:
            break

    return task_data
```

- Dataset composé de :
  - Images
  - Labels
  - Angles de rotation
- load_rmnist_task_data()
  - Met les données dans un tableau
- Modification train() et test()
  - Extractions des données du tableau dans une boucle

Modèle à circuit unique

- Utilisation d'un layer personnalisé du code d'origine

```
net = torch.nn.Sequential(
            layers.ShareConv2d(1, 32, kernel_size=3),
            nn.ReLU(),
            nn.Flatten(),
            layers.ShareLinearFull(32 * 26 * 26, 10)
).to(device)
```

- Couche à poids partagée spécifique pour les images en 2D

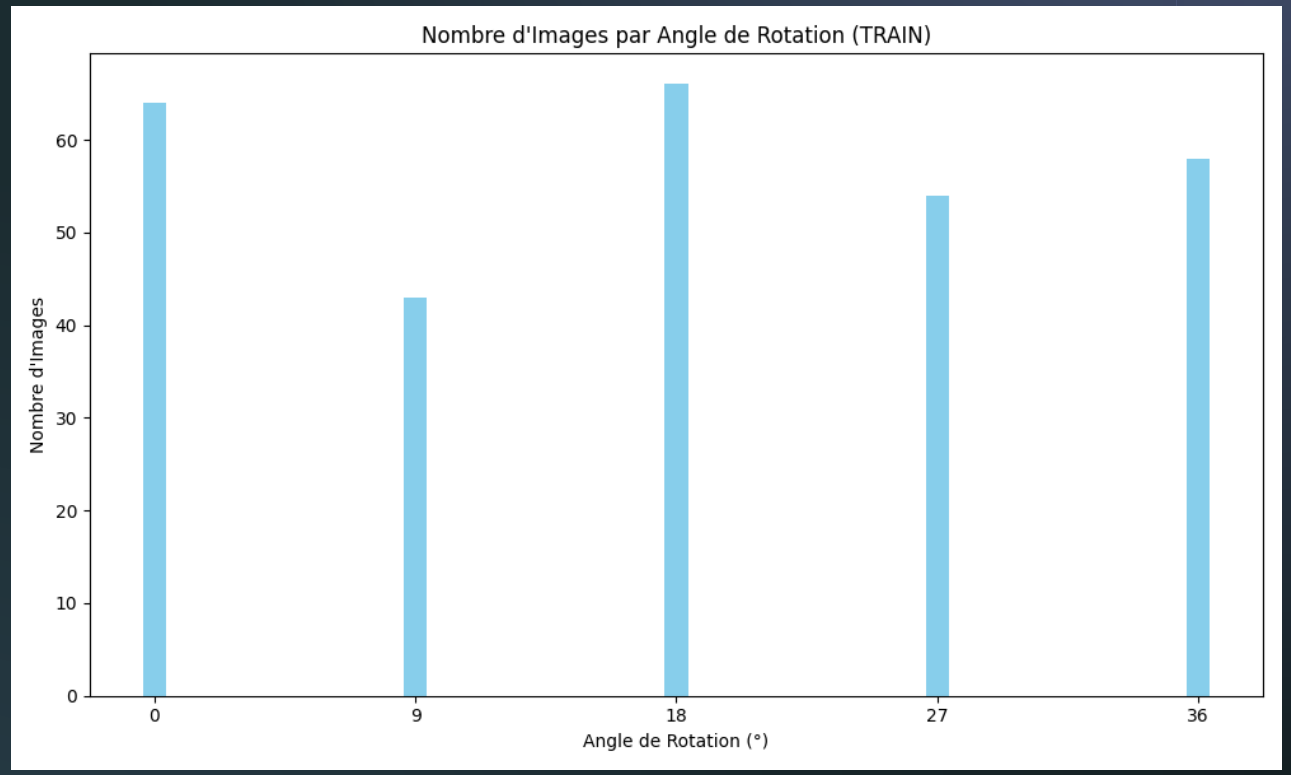
Modèle à circuit dédié

- Un circuit par angle de rotation
- Couche d'alignement en sortie

```
circuits = nn.ModuleList()
        for _ in range(num_tasks):
            circuit = nn.Sequential(
                layers.ShareConv2d(1, 32, kernel_size=3), nn.ReLU(), nn.Flatten())
            circuits.append(circuit)
        align_layer = layers.ShareLinearFull(32 * 26 * 26 * num_tasks, 10)
```

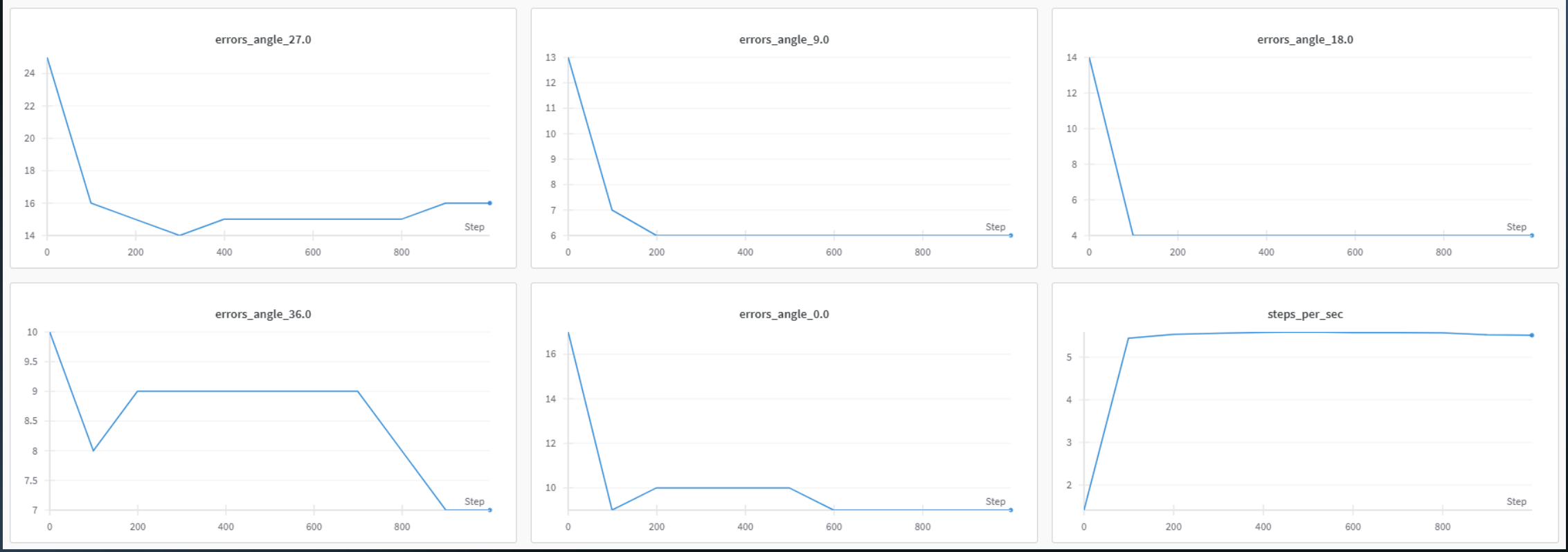
```
net = DedicatedNetwork(circuits, align_layer).to(device)
```

Visualisation des angles



- Modification train() et test():
  - Génération histogramme
  - Répartition par angle de rotation
  - Nombre d’erreurs de prédiction par angle

Visualisation des angles

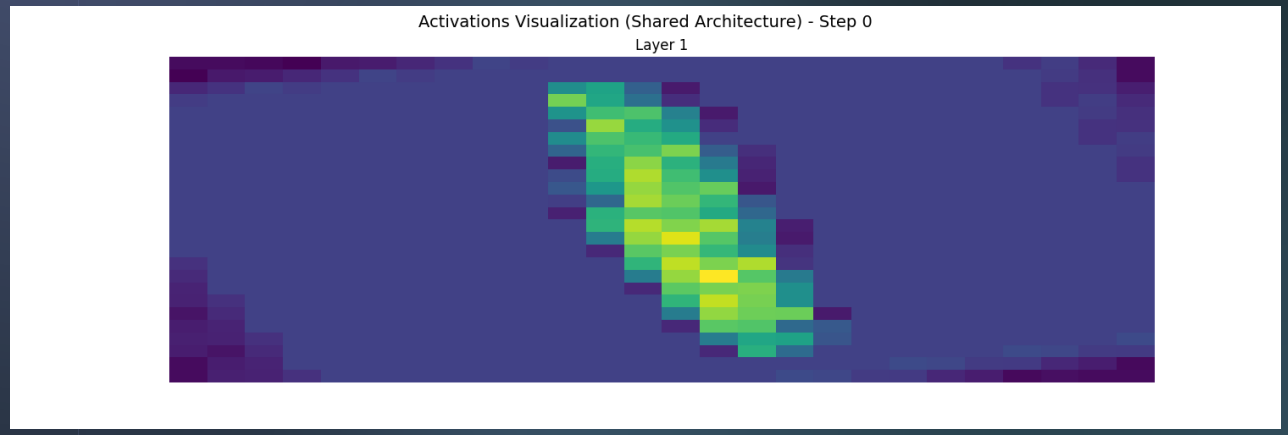


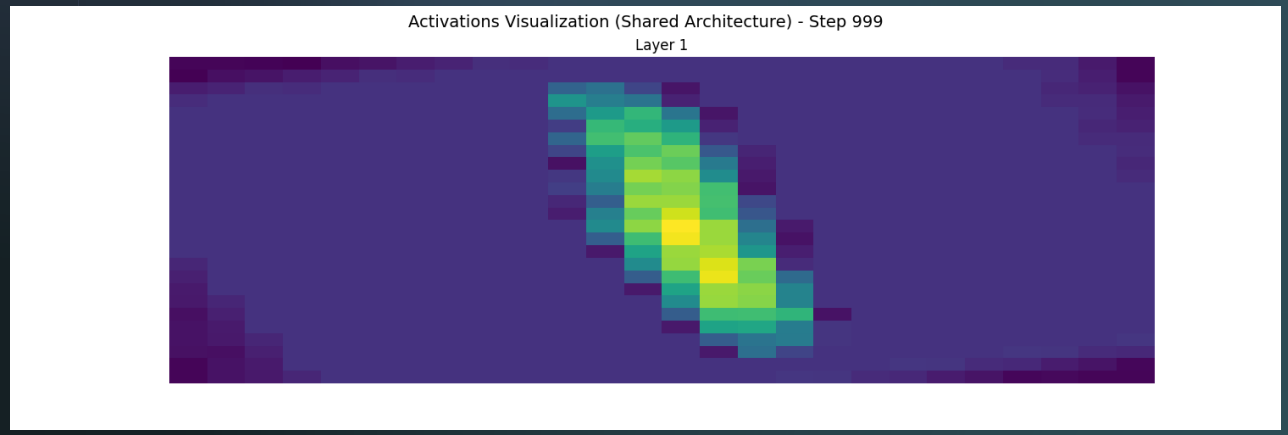
Visualisation weight sharing

- Fonction de visualisation: visualize_weights()
- poids convolutifs du modèle
- Filtre: matrice 2D
- Heatmap

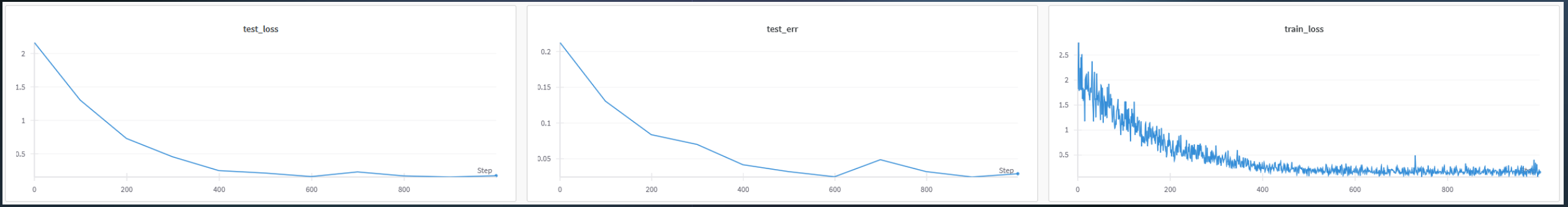
Visualisation weight sharing

- Activation intermédiaire
- Caractéristiques détectées par chaque couche du réseau:
  - Ligne, courbe, autres…
- Fonction de visualisation: visualize_activations()
- Heatmap

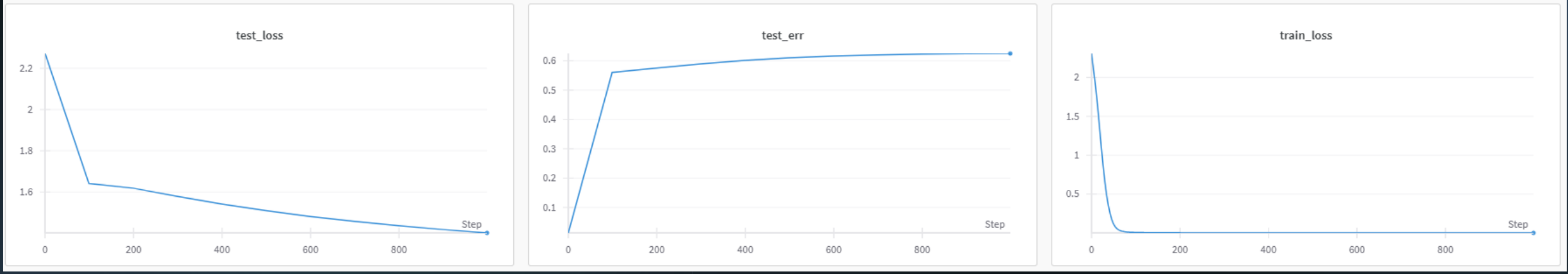




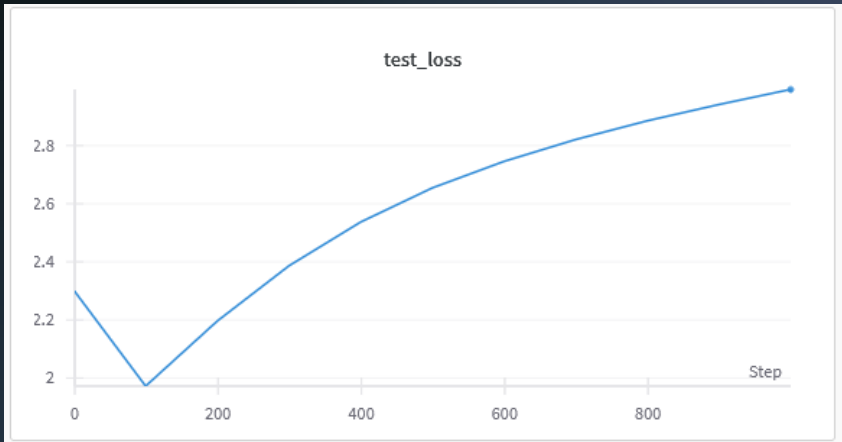
Résultats données initiales

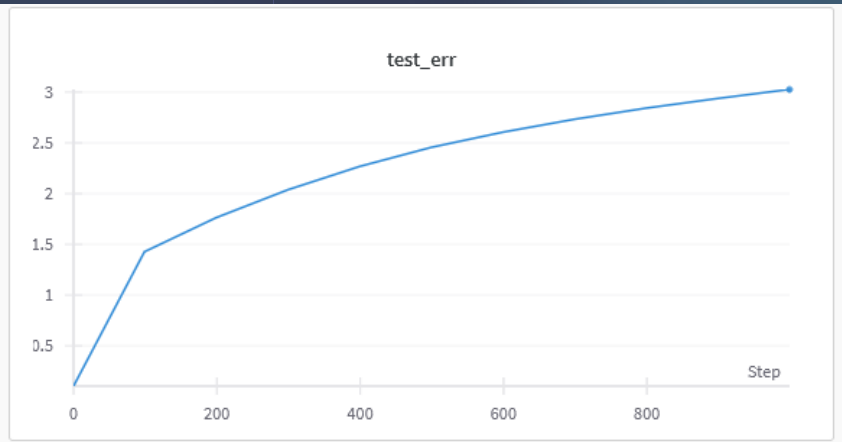


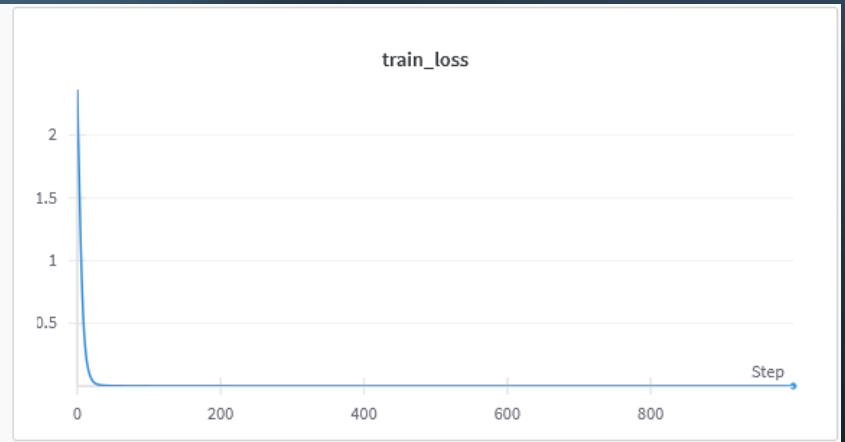
Résultats circuit unique



Résultats circuit dédié







Conclusion

- Adaptation au dataset Rotated MNIST : ✓
- Implémentation architecture à circuit unique et partagé : ✓
- Implémentation architecture à circuit dédié avec alignement : à améliorer
- Modèle adaptable aux données et aux problèmes : ✓