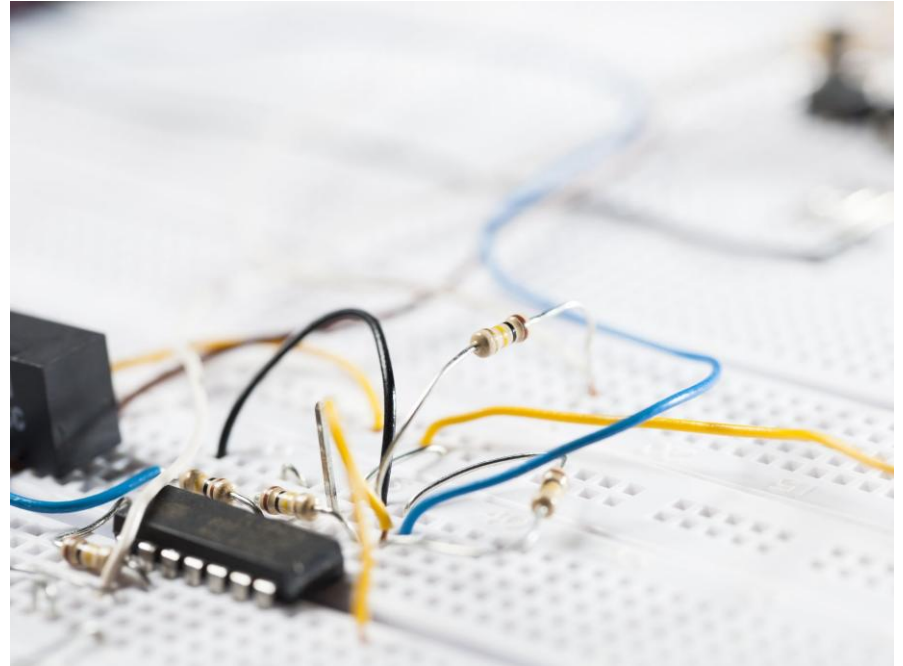


Ottimizzazione di TinyViT per Sistemi Embedded tramite Distillazione, Pruning e Quantizzazione

Autore: Davide Iannella

Progetto Deep Learning

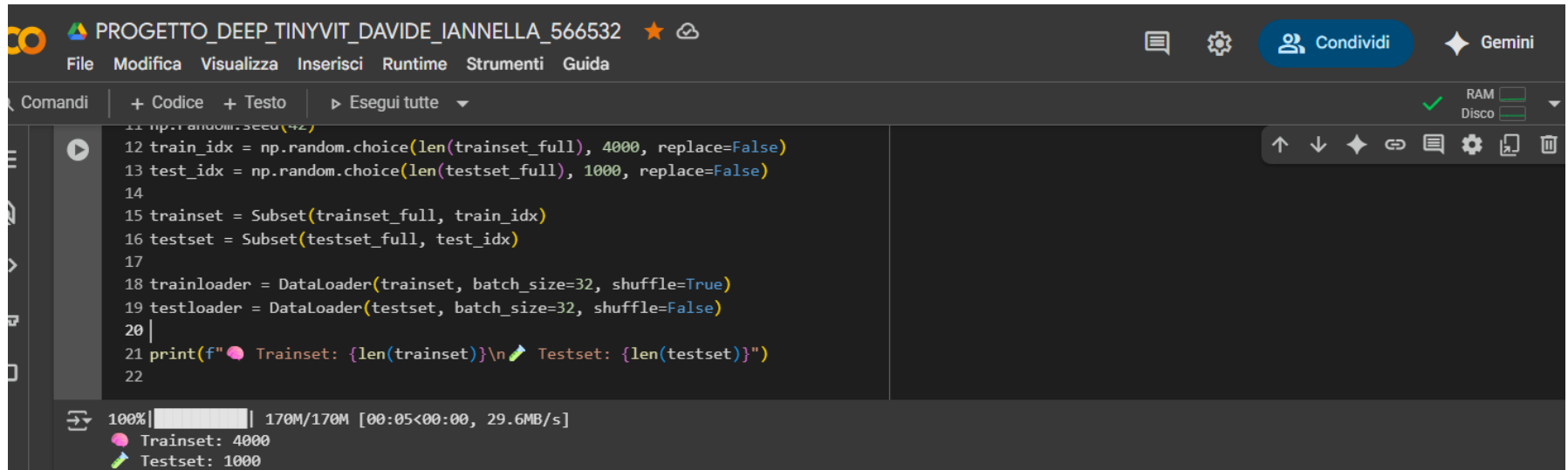




OBIETTIVO: OTTIMIZZARE UN
MODELLO TRANSFORMER (TINYVIT)
PER DISPOSITIVI EMBEDDED



METODI: DISTILLAZIONE, PRUNING E
QUANTIZZAZIONE.



The screenshot shows a JupyterLab environment with a dark theme. The top bar includes the project name "PROGETTO_DEEP_TINYVIT_DAVIDE_IANNELLA_566532" and a "Condividi" button. The left sidebar has tabs for "Comandi", "+ Codice", "+ Testo", and "Esegui tutte". The main area displays Python code for splitting a dataset into training and testing sets using NumPy and PyTorch's Subset and DataLoader classes. The code is as follows:

```
11 np.random.seed(42)
12 train_idx = np.random.choice(len(trainset_full), 4000, replace=False)
13 test_idx = np.random.choice(len(testset_full), 1000, replace=False)
14
15 trainset = Subset(trainset_full, train_idx)
16 testset = Subset(testset_full, test_idx)
17
18 trainloader = DataLoader(trainset, batch_size=32, shuffle=True)
19 testloader = DataLoader(testset, batch_size=32, shuffle=False)
20
21 print(f"🌸 Trainset: {len(trainset)}\n🌿 Testset: {len(testset)}")
22
```

At the bottom, a progress bar shows 100% completion. Below it, the output of the print statement is displayed: "🌸 Trainset: 4000" and "🌿 Testset: 1000". The status bar at the very bottom indicates "170M/170M [00:05<00:00, 29.6MB/s]" and shows RAM and Disco usage.

Dataset: CIFAR-10

10 classi, immagini 32x32 RGB, ridimensionate a 224x224 per compatibilità ViT



```
1 teacher = torchvision.models.resnet18(weights='IMAGENET1K_V1')
2 teacher.fc = nn.Linear(teacher.fc.in_features, 10)
3 teacher = teacher.to(device)
4 teacher.eval()
5
6 student = timm.create_model('vit_tiny_patch16_224', pretrained=True)
7 student.head = nn.Linear(student.head.in_features, 10)
8 student = student.to(device)
9
10 summary(student, input_size=(3, 224, 224), device=str(device))
11
```



Downloading: "<https://download.pytorch.org/models/resnet18-f37072fd.pth>" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|██████████| 44.7M/44.7M [00:00<00:00, 126MB/s]
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
model.safetensors: 100% ██████████ 22.9M/22.9M [00:00<00:00, 71.3MB/s]

Modello Teacher: ResNet-18 pretrained su
ImageNet

Fine-tuned per 10 classi.



```
1 teacher = torchvision.models.resnet18(weights='IMAGENET1K_V1')
2 teacher.fc = nn.Linear(teacher.fc.in_features, 10)
3 teacher = teacher.to(device)
4 teacher.eval()
5
6 student = timm.create_model('vit_tiny_patch16_224', pretrained=True)
7 student.head = nn.Linear(student.head.in_features, 10)
8 student = student.to(device)
9
10 summary(student, input_size=(3, 224, 224), device=str(device))
11
```



Downloading: "<https://download.pytorch.org/models/resnet18-f37072fd.pth>" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth [100%] 44.7M/44.7M [00:00<00:00, 126MB/s]

/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.

warnings.warn(
model.safetensors: 100% 22.9M/22.9M [00:00<00:00, 71.3MB/s]

Modello Student:
TinyViT
(vit_tiny_patch16_224)

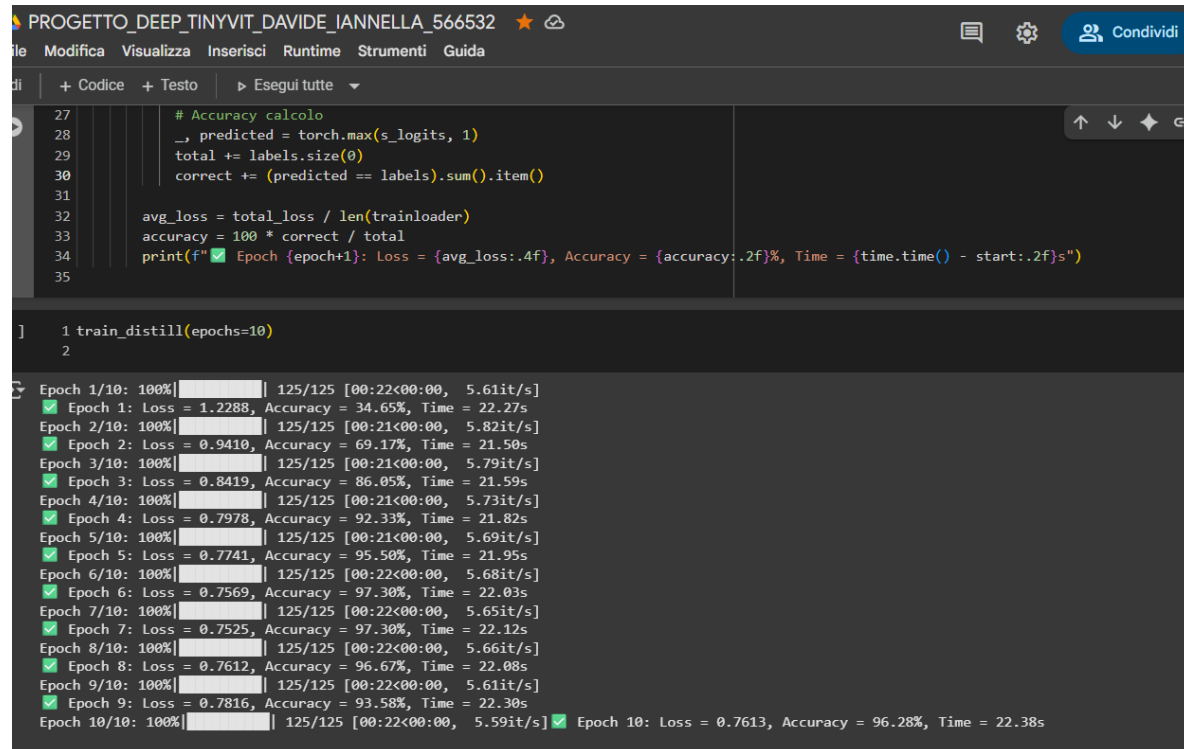
Piccolo ViT pre-
addestrato, adattato a
10 classi.

Distillazione: conoscenza
trasferita dal teacher al
student

Loss = KL Divergence + 0.5
* CrossEntropy.

```
loss = kd_loss(torch.log_softmax(s_logits, dim=1),  
               torch.softmax(t_logits, dim=1)) + 0.5 * ce_loss(s_logits, labels)
```

Ottimizzazione:
Adam, lr=3e-4,
batch=32, 10 epoche
su subset CIFAR10
(4000 train, 1000
test).



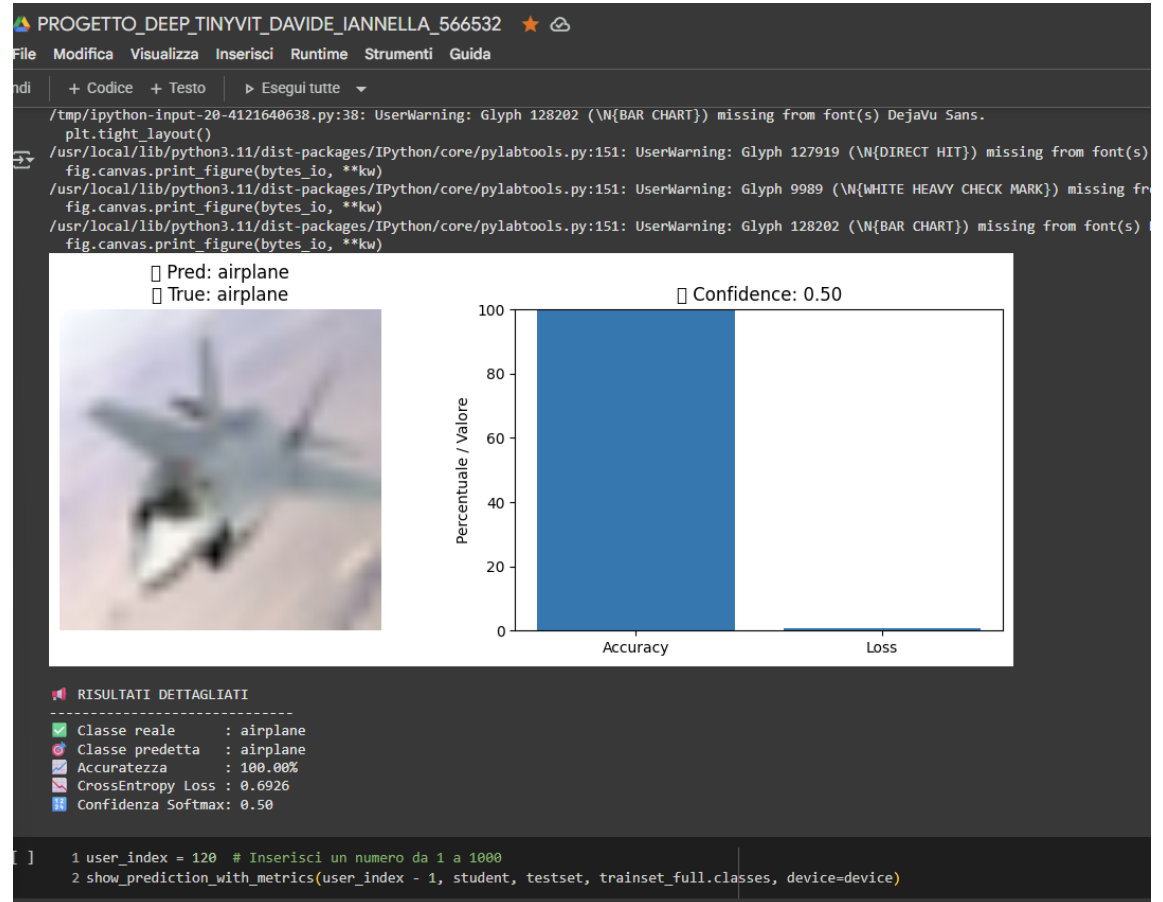
The screenshot shows a Jupyter Notebook titled "PROGETTO_DEEP_TINYVIT_DAVIDE_IANNELLA_566532". The code in the cell defines an accuracy calculation function and a training loop. The output shows the progress of 10 epochs, with loss decreasing and accuracy increasing over time.

```
27 # Accuracy calcolo
28 _, predicted = torch.max(s_logits, 1)
29 total += labels.size(0)
30 correct += (predicted == labels).sum().item()
31
32 avg_loss = total_loss / len(trainloader)
33 accuracy = 100 * correct / total
34 print(f"✅ Epoch {epoch+1}: Loss = {avg_loss:.4f}, Accuracy = {accuracy:.2f}%, Time = {time.time() - start:.2f}s")
35
```

```
] 1 train_distill(epochs=10)
2
```

Epoch 1/10: 100%|██████████| 125/125 [00:22<00:00, 5.61it/s]
✅ Epoch 1: Loss = 1.2288, Accuracy = 34.65%, Time = 22.27s
Epoch 2/10: 100%|██████████| 125/125 [00:21<00:00, 5.82it/s]
✅ Epoch 2: Loss = 0.9410, Accuracy = 69.17%, Time = 21.50s
Epoch 3/10: 100%|██████████| 125/125 [00:21<00:00, 5.79it/s]
✅ Epoch 3: Loss = 0.8419, Accuracy = 86.05%, Time = 21.59s
Epoch 4/10: 100%|██████████| 125/125 [00:21<00:00, 5.73it/s]
✅ Epoch 4: Loss = 0.7978, Accuracy = 92.33%, Time = 21.82s
Epoch 5/10: 100%|██████████| 125/125 [00:21<00:00, 5.69it/s]
✅ Epoch 5: Loss = 0.7741, Accuracy = 95.50%, Time = 21.95s
Epoch 6/10: 100%|██████████| 125/125 [00:22<00:00, 5.68it/s]
✅ Epoch 6: Loss = 0.7569, Accuracy = 97.30%, Time = 22.03s
Epoch 7/10: 100%|██████████| 125/125 [00:22<00:00, 5.65it/s]
✅ Epoch 7: Loss = 0.7525, Accuracy = 97.30%, Time = 22.12s
Epoch 8/10: 100%|██████████| 125/125 [00:22<00:00, 5.66it/s]
✅ Epoch 8: Loss = 0.7612, Accuracy = 96.67%, Time = 22.08s
Epoch 9/10: 100%|██████████| 125/125 [00:22<00:00, 5.61it/s]
✅ Epoch 9: Loss = 0.7816, Accuracy = 93.58%, Time = 22.30s
Epoch 10/10: 100%|██████████| 125/125 [00:22<00:00, 5.59it/s] ✅ Epoch 10: Loss = 0.7613, Accuracy = 96.28%, Time = 22.38s

Risultati
Distillazione:
Accuratezza
test > 90%,
perdita
ridotta,
training
stabile.



Pruning:

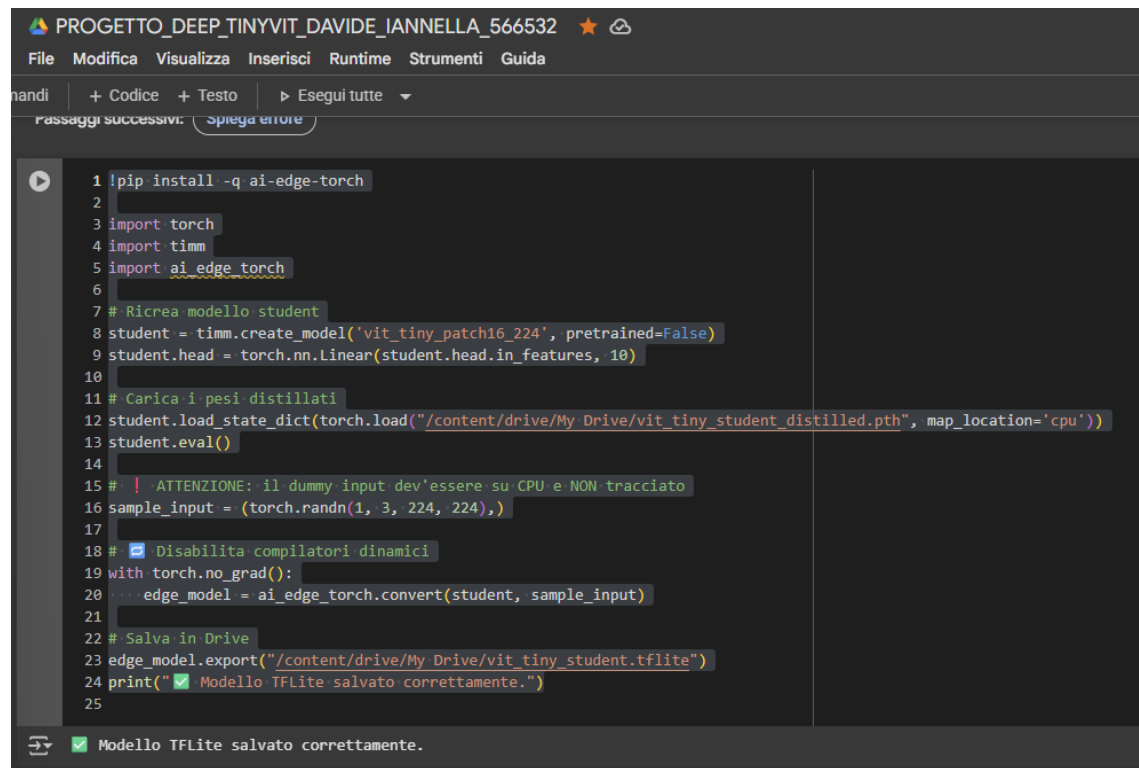
Disattivazione drop-out e moduli ridondanti interni a TinyViT.



```
1 for name, module in student.named_modules():  
2     if hasattr(module, 'attn_drop'):  
3         module.attn_drop.p = 0.0  
4
```

Effetto Pruning:
Accuratezza conservata,
modello più leggero e
veloce.

Quantizzazione:
uso libreria ai-edge-
torch per
esportazione a
TFLite
Input dummy CPU,
esportazione
edge_model.



```
PROGETTO_DEEP_TINYVIT_DAVIDE_IANNELLA_566532 ★ ☁
File Modifica Visualizza Inserisci Runtime Strumenti Guida
nandi + Codice + Testo ▶ Esegui tutte ▼
Passaggi successivi: Spiega errore


1 !pip install -q ai-edge-torch
2
3 import torch
4 import timm
5 import ai_edge_torch
6
7 # Ricrea modello student
8 student = timm.create_model('vit_tiny_patch16_224', pretrained=False)
9 student.head = torch.nn.Linear(student.head.in_features, 10)
10
11 # Carica i pesi distillati
12 student.load_state_dict(torch.load("/content/drive/My Drive/vit_tiny_student_distilled.pth", map_location='cpu'))
13 student.eval()
14
15 # ! ATTENZIONE: il dummy input dev'essere su CPU e NON tracciato
16 sample_input = (torch.randn(1, 3, 224, 224),)
17
18 # Disabilita compilatori dinamici
19 with torch.no_grad():
20     edge_model = ai_edge_torch.convert(student, sample_input)
21
22 # Salva in Drive
23 edge_model.export("/content/drive/My Drive/vit_tiny_student.tflite")
24 print("✅ Modello TFLite salvato correttamente.")
25

✅ Modello TFLite salvato correttamente.
```

Verifica Modello
TFLite:
accuratezza
coerente,
inference su
CIFAR-10
Script Python +
libreria TFLite +
matplotlib.

```
PROGETTO_DEEP_TINYVIT_DAVIDE_IANNELLA_566532 ★ ☁
File Modifica Visualizza Inserisci Runtime Strumenti Guida
andi + Codice + Testo ▶ Esegui tutte ▼
46
47 # 🔹 Visualizza immagine e risultato
48 plt.imshow(np.transpose(image.numpy(), (1, 2, 0)) * 0.5 + 0.5)
49 plt.axis('off')
50 plt.title(f"🔗 Predetto: {pred_label} | ✅ Reale: {true_label}")
51 plt.show()
52
53 # 🔊 Stampa dettagli
54 print(f"✅ Predizione TFLite: {pred_label}")
55 print(f"🔗 Classe reale : {true_label}")
56 print(f"📊 Probabilità max : {np.max(output):.2f}")
57
```

🖼️ ☐ Predetto: bird | ☐ Reale: bird



✅ Predizione TFLite: bird
🔗 Classe reale : bird
📊 Probabilità max : 1.84



Criticità Distillazione: difficile
bilanciamento tra CE e KLDivLoss



Necessario tuning empirico.



Criticità Pruning: rimozione layer
delicata, rischio underfitting



Scelta: solo dropout disattivati.

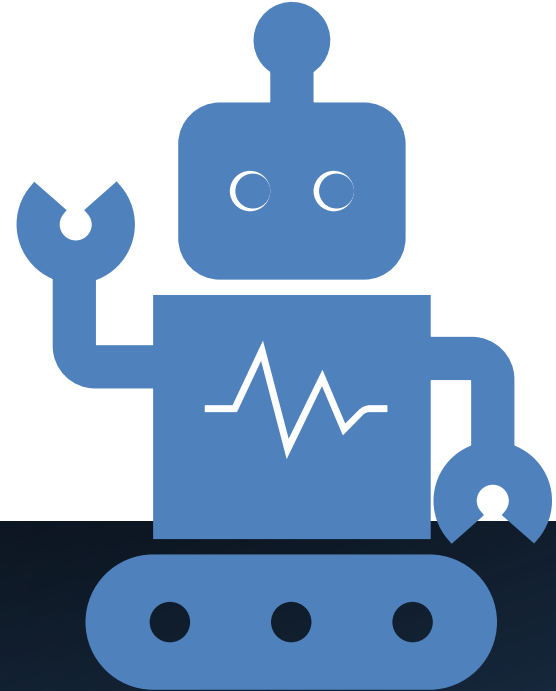
Criticità Quantizzazione: TFLite
richiede CPU, incompatibilità GPU

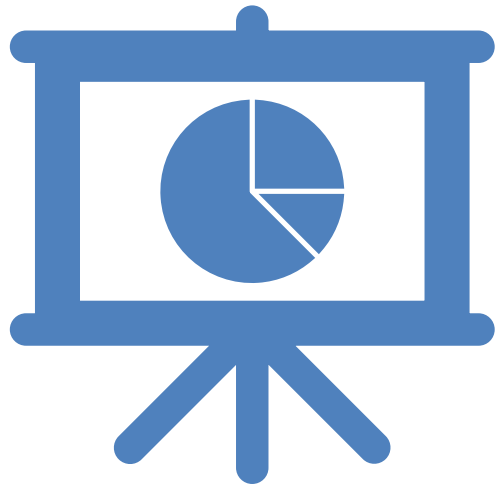


Formati e export delicati.

Inoltre TinyViT non supporta
pienamente la quantizzazione
int8: il modello TFLite è float32.

Embedded Ready:
modello TFLite usabile
in Android, Raspberry,
microcontrollori
avanzati.





- Visualizzazione Risultati:
- Funzione matplotlib con grafico loss/confidenza + immagini predette.

Struttura Codice:

1. Training distillazione

2. Pruning

3. Esportazione

4. Verifica TFLite

Futuri sviluppi:



- Addestramento full CIFAR10

- Edge deployment (Raspberry Pi)

- Quantizzazione int8.

Grazie per l'attenzione