# FixMatch com CIFAR-10

## Implementação e avaliação

Kauan Mariani Ferreira
Pedro Henrique Coterli
Samuel Corrêa Lima

FGV

# Implementação

- Função **get_ssl_indices**: Divide os índices do dataset em "rotulados" e "não rotulados".
- Datasets **CIFAR10Labeled** (rotulado) e **CIFAR10Unlabeled** (não rotulado).
- Rede **Resnet** composta por blocos **BasicBlock** para servir como o modelo base.

# Função de perda

$$\ell_s = \frac{1}{B} \sum_{b=1}^{B} \mathrm{H}(p_b, \alpha(x_b)) \; \{\textit{Cross-entropy loss for labeled data}\}$$

$$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}\{\max(q_b) > \tau\} \, \mathrm{H}(\arg\max(q_b), p_{\mathrm{m}}(y \mid \mathcal{A}(u_b))$$

$$\{\textit{Cross-entropy loss with pseudo-label and confidence for unlabeled data}\}$$

$$\textbf{return } \ell_s + \lambda_u \ell_u$$

# Otimizador

- Buscamos uma fidelidade ao paper, e de acordo com o ablation study (secção 5 do paper), o melhor otimizador é o SGD com nesterov, e foi esse o otimizador que utilizamos.

# Exponential Moving Average

- Outro ponto que foi importante no paper é a utilização de uma média móvel exponencial para a previsão. Ao invés de utilizar apenas o output esperado, para a previsão é feita a utilização de uma média móvel exponencial dos últimas previsões .

# Weight Decay

- Implementamos o “decaimento” da learning rate como no paper, por cosseno.

$$\eta \cdot \cos\left(\frac{7 \cdot \pi \cdot k}{16 \cdot K}\right)$$

- Onde eta é a Lr inicial, k é o passo que estamos e K é o total de passos que teremos

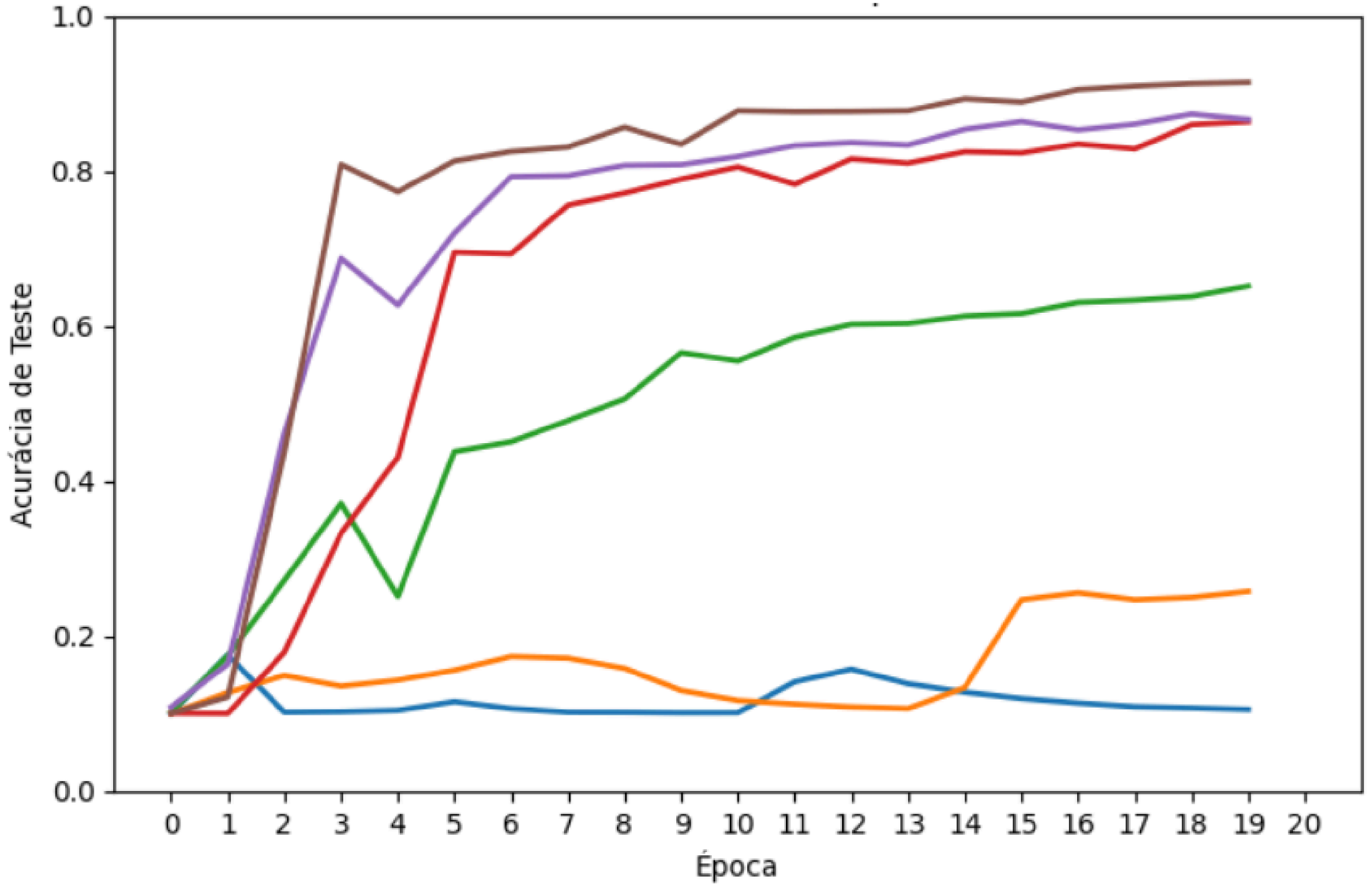
First, as mentioned above, we find that regularization is particularly important. In all of our models and experiments, we use simple weight decay regularization. We also found that using the Adam optimizer [22] resulted in worse performance and instead use standard SGD with momentum [50, 40, 34]. We did not find a substantial difference between standard and Nesterov momentum. For a learning rate schedule, we use a cosine learning rate decay [28] which sets the learning rate to $\eta \cos\left(\frac{7\pi k}{16K}\right)$ where $\eta$ is the initial learning rate, $k$ is the current training step, and $K$ is the total number of training steps. Finally, we report final performance using an exponential moving average of model parameters.

# Novo teste sugerido:

- Em princípio, utilizamos  uma resnet treinada do 0 para aprender os dados, mas notamos que poderíamos nos beneficiar de transfer learning para uma representação mais eficiente.  Assim, carregamos os pesos de uma resnet18 do pytorch e então treinamos e avaliamos o modelo

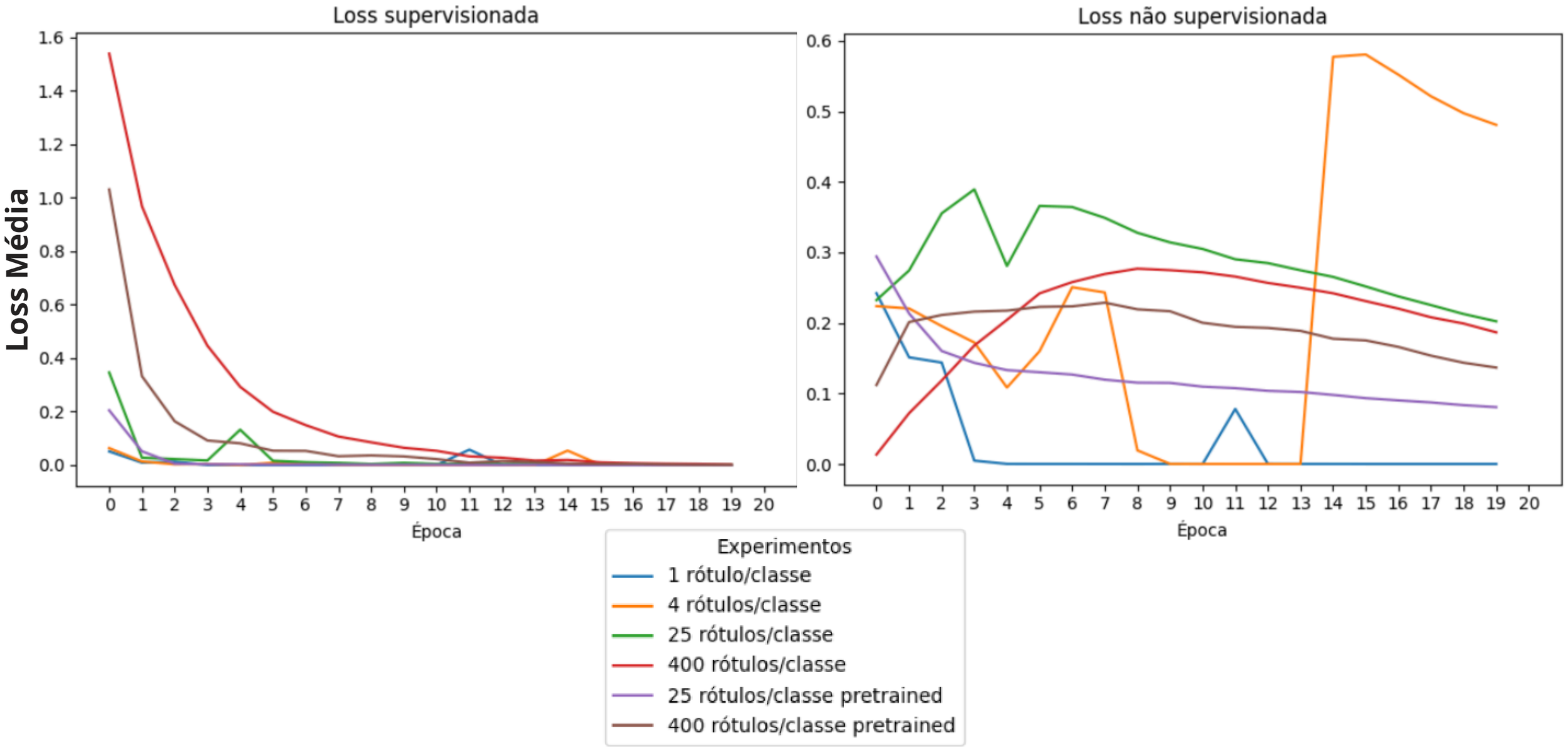
# Resultados dos Experimentos

# Acurácia do FixMatch por rótulos

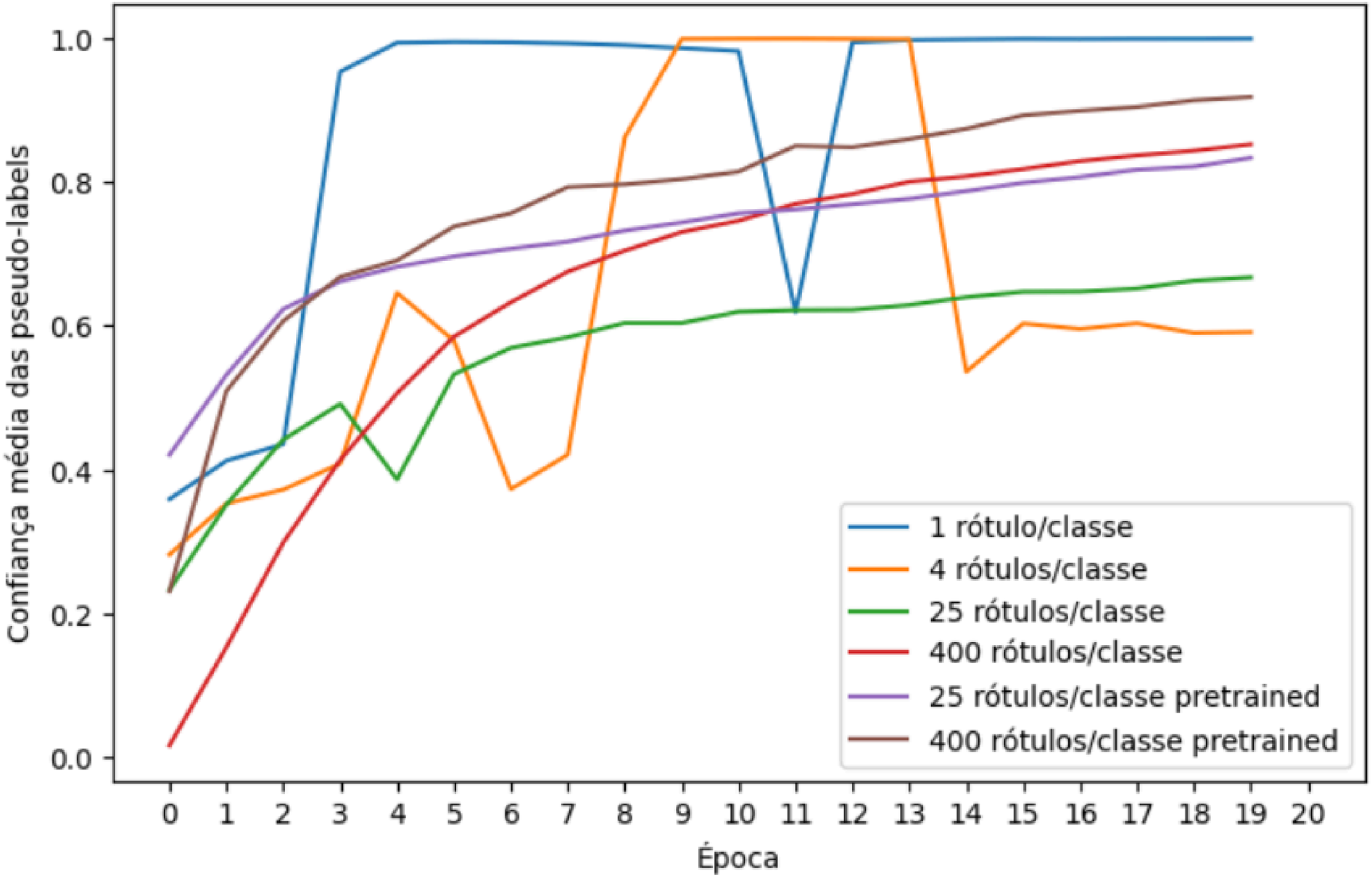



| | Acurácia final (%) |
|---|---|
| **1 rótulo/classe** | 10.51 |
| **4 rótulos/classe** | 25.77 |
| **25 rótulos/classe** | 65.16 |
| **400 rótulos/classe** | 86.35 |
| **25 rótulos/classe pretrained** | 86.64 |
| **400 rótulos/classe pretrained** | 91.43 |

# Comparação dos tipos de Loss



# Evolução da confiança das pseudo-labels



# Conclusão:

Os resultados mostram que o desempenho do FixMatch depende fortemente da quantidade de rótulos e do pré-treinamento. Com poucos rótulos, o modelo quase não aprende, mas a partir de 25 por classe há grande melhora, especialmente quando há pré-treinamento, que eleva a acurácia para cerca de 90%.

A loss supervisionada converge rapidamente em todos os casos, enquanto a não supervisionada é instável com poucos rótulos e se torna mais consistente com o pré-treinamento.

Já o gráfico de confiança revela que, com poucos rótulos, o modelo se torna superconfiante em previsões incorretas, enquanto o aumento de rótulos e o pré-treinamento promovem aprendizado mais equilibrado e pseudo-rótulos mais confiáveis.

# Obrigado!

# Implementação