**ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO**

**Departamento de Engenharia de Computação e Sistemas Digitais**



**PCS 5022 - Redes Neurais e Aprendizado Profundo**

Prof. Artur Jordão

# 1 Fundamentos Básicos de Machine Learning

1. O código abaixo apresenta um problema teórico grave. Qual? Como resolver? Crie um modelo preditivo (i.e., OLS, SVM, XGBoost) e analise o comportamento da performance preditiva do modelo antes e após resolver o problema.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

n_samples = 1000
n_features = 2
n_classes = 5
X = np.random.rand(n_samples, n_features)
y = np.random.randint(0, n_classes, size=n_samples)
y = np.eye(n_classes)[y]

mean = np.mean(X, axis=0)
std = np.std(X, axis=0)
X = (X - mean) / std

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33,
random_state=42)

model = ...
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy_score(y_test, y_pred)
```

Código 1: Exercício 1

2. Gere dados aleatórios com dimensão $n \times m$ para $k$ categorias (classes) e $m = 2$. A partir dos dados gerados realize as seguintes tarefas. (i) Calcule a média das amostras; (ii) Calcule a média das amostras de cada classe; (iii) Para cada classe, identifique qual a amostra mais distante da amostra média da classe. Expresse formalmente a solução dos itens (i)-(iv).

3. Gere dados aleatórios com dimensão $n \times m$, com $m = 2$. A partir desses dados mostre o espaço de características, destacando a amostra média geral e a amostra média por classe.

4. Faça o mesmo que o exercício anterior, porém agora utilize $m > 4$. Como mostrar o espaço de características para valores de $m$ arbitrariamente grandes?

5. Um problema envolvendo a solução analítica da regressão linear, $(X^T X)^{-1} X^T Y$, é que os dados de treinamento não podem apresentar mais features ($m$) do que amostras ($n$) – *the sample size problem*. Como resolver esse problema sem utilizar a otimização de *gradient descent* (ou métodos iterativos de otimização)?

6. Suponha um cenário hipotético em que só podemos aprender um modelo usando um determinado número de amostras $k$, onde $k \ll n$. Como lidar com essa restrição evitando uma solução trivial de amostrar exemplos aleatoriamente?
   Leitura recomendada para o exercício: [31]

# 2 FeedForward Networks

1. Assuma dois valores inteiros $n$ e $k$. A partir desses valores, crie uma rede MLP com $n$ camadas, onde a $i$-ésima camada ($i \in \{1, 2, 3, \ldots, n\}$) possui $k^i$ neurônios.

2. Uma similaridade entre projeções lineares (ex., predição de uma regressão linear ou transformação via *Principal Components Analysis* – PCA) e uma rede neural com somente uma camada é que podemos formular a predição como $xW^T + b$. Tecnicamente, podemos utilizar essa similaridade para modelar predições de modelos lineares como redes neurais. Como? Tente realizar esse processo para uma projeção linear simples (ex. PCA). Observe como a matriz de projeção é armazenada: matriz coluna – $k \times 1$ – ou matriz linha – $1 \times k$. Note também as possíveis transformações que podem ser realizadas nos dados antes de projetá-los. O código abaixo ilustra com utilizar o PCA disponível no sklearn.

```
from sklearn.decomposition import PCA
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=10000, n_features=3000...)

pca = PCA(n_components=2)
pca.fit(X)
X_latent = pca.transform(X)
w = pca.components_
...
```

Código 2: Exercício 2

3. Podemos dizer que modelos superparametrizados podem classificar dados com rótulos aleatórios. Desenvolva um setup experimental para demonstrar (mesmo que parcialmente) essa afirmação. Tal característica corresponde a uma propriedade positiva ou negativa desses modelos?
   Leitura recomendada para o exercício: [53, 32]

# 3 Basics Hyperparameters

1. Formule o grafo computacional das seguintes expressões: $y = 2 * (a - b) + c$; $y = x^T w + b$ e $y = (\frac{x-\mu}{\sigma}) * \gamma + \delta$.

2. A partir do grafo computacional abaixo, indique qual a derivada parcial de $\frac{\partial g}{\partial b}$ (isto é, como alterações em $b$ afetam $g$).

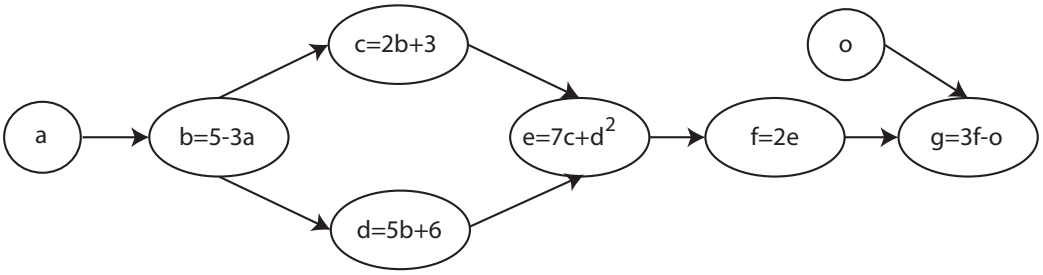


Figure 1: Grafo computacional para o Exercício 2

3. (⋆) Proponha uma técnica *learning rate scheduler*.

4. (⋆) Proponha uma função de ativação.
Leitura recomendada para o exercício: [39].

5. (⋆) Proponha uma função de erro. A função de erro pode ser uma combinação de operadores matemáticos primitivos.
Leitura recomendada para o exercício: [27, 28].

6. Elabore uma técnica para explorar um espaço $n$ dimensional de hiperparâmetros (ex. *grid search*). Seu espaço deve conter, pelo menos, os seguintes hiperparâmetros: (i) otimizador, (ii) inicialização e (iii) *learning rate* inicial. Em cada eixo do espaço (isto é, cada hiperparâmetro) deve existir pelo menos duas opções. Visualizando essa questão da perspectiva de *feature space*, seria possível predizer a performance preditiva do modelo para valores de hiperparâmetros não observáveis (dica: pense como seria a composição das matrizes $X$ e $Y$)?

# 4 Deep Learning

1. Frequentemente em *Deep Learning*, a quantidade de parâmetros é um indicativo da capacidade de expressividade do modelo. A partir dessa observação, uma forma de controlar a capacidade de um modelo é aumentar o número de neurônios ou a quantidade de camadas (paradigma *we need to go deeper*). Assuma um valor inteiro $n$ indicando a quantidade de neurônios de duas camadas ocultas compondo uma rede neural. Construa outra arquitetura distribuindo esses $n$ neurônios em $k$ camadas. Por exemplo, a partir de uma rede com

$n = 2048$ (largura $\{2048, 2048\}$) podemos criar outra com o mesmo número de neurônios usando $k = 4$ ($\{1024, 1024, 1024, 1024\}$) ou $k = 8$ ($\{512, 512, 512, 512, 512, 512, 512, 512\}$). Qual dos dois modelos é mais eficiente em termos de latência[1], armazenamento e número total de parâmetros?
Leitura recomendada para o exercício: [41, 11, 3]

2. Podemos entender a transformação aplicada por uma camada $i$ como a representação interna dessa camada para os dados. A partir dessa descrição, como podemos mensurar a similaridade entre representações internas de duas camadas distintas da mesma rede? E de redes neurais diferentes? O código abaixo sumariza como obter a representação interna de uma camada $i$ usando um modelo pré-definido.
Leitura recomendada para o exercício: [26, 35, 36, 5, 22, 23]

```
model_ = keras.models.Model(model.input,
model.get_layer(index=i).output)
internal_representation = model_.predict(X)
```

3. Combine a representação interna de diferentes camadas e aplique essa combinação ($X = \bigcup_{i=1}^{L} X_i$) a um classificador simples.

4. Considerando uma rede neural qualquer, elabore um experimento para ilustrar a separabilidade fornecida pela representação interna das diferentes camadas compondo o modelo.

5. De acordo com o trabalho de Zhang et al. [54], as camadas em modelos profundos podem ser categorizadas em robustas e críticas. Camadas robustas são aquelas que, após o treinamento, podem ser reinicializadas a sua inicialização original sem degradar (ou degradando marginalmente) a habilidade preditiva do modelo. Camadas críticas, por outro lado, são sensível a reinicialização. Elabore um setup experimental simples para demonstrar essa observação. Ao final, mostre quais foram as camadas críticas e as robustas. Observação importante: as conclusões de Zhang et al. [54] são aplicáveis às arquiteturas residuais que serão abordadas posteriormente no curso.
Leitura recomendada para o exercício: [54, 33]

# 5 Regularization

1. Construa uma arquitetura de rede neural simples e a regularize-a utilizando o mecanismo de ensemble. Defina formalmente a estratégia de ensemble adotada.

2. Períodos críticos correspondem a um fenômeno relacionado à efetividade da regularização na dinâmica do treinamento. De acordo com trabalhos anteriores, tal fenômeno ocorre nas épocas iniciais de treinamento. A partir de uma rede neural qualquer, elabore um setup

[1] Latência refere-se ao tempo que um modelo leva para predizer a resposta a partir de uma amostra ou conjunto de amostras (tempo de *foward*).

experimental para inspecionar se períodos críticos emergem durante o treinamento dessa arquitetura.
Leitura recomendada para o exercício: [9, 24, 25]

3. ($\star$) Elabore uma técnica para identificar quando (época) períodos críticos terminam.

4. Conforme visto em aula, estudos mostraram a possibilidade de aprender somente os parâmetros das camadas de Batch Normalization. Elabore um treinamento para demonstrar a eficácia dessa estratégia (isto é, se um modelo treinado seguindo essa ideia obtém habilidade preditiva não trivial).
Leitura recomendada para o exercício: [8, 1]

# 6 Convolutional Networks

1. Selecione uma rede neural convolucional (por exemplo, a partir do keras applications) e, a partir desta rede, escolha $n$ camadas para realizar extração de *features*. Para cada camada $i \in n$, projete suas respectivas *features* em um espaço de duas dimensões (ex., usando PCA ou outro método de redução de dimensionalidade). Em seguida, analise se alguma camada proporciona uma melhor separação dos dados.

2. Um dos maiores gargalos computacionais envolvendo redes convolucionais está relacionado à resolução espacial (largura $\times$ altura – $W \times H$) da imagem de entrada. Interessantemente, estudos confirmaram a possibilidade de adaptar uma rede pré-treinada para realizar a inferência (*forward*) em uma resolução diferente da utilizada durante o treinamento. Por exemplo, Touvron et al. [42] observou que é possível treinar uma rede convolucional em determinada escala e aplicar uma escala maior somente nas imagens de teste; desta forma, reduzindo notavelmente o tempo de treinamento/*fine-tuning*.

   Formalmente, seja $\mathcal{F}(\cdot^{224\times224}, \theta)$ uma rede convolucional treinada a partir de amostras com resolução $224 \times 224$. Proponha uma solução que permita utilizar $\theta$ sem qualquer alteração para predizer amostras de resoluções arbitrárias ($x \in \mathbb{R}^{W\times H}$ – $\mathcal{F}(\cdot^{W\times H}, \theta)$). Qual componente da arquitetura possibilita a solução? Observe que o problema não corresponde a alterar a resolução da entrada. Para facilitar a implementação utilize a arquitetura ResNet50 disponível no github da disciplina e os pesos treinados no ImageNet disponíveis neste link.
   Leitura recomendada para o exercício: [42]

3. ($\star$) Seguindo o exercício acima, proponha um modelo para perver qual a melhor resolução para realizar o *forward* de uma imagem $x$ em uma rede convolucional. Por exemplo, considere um modelo $\mathcal{P}(x, \cdot)$ que estima qual a melhor resolução para classificar $x$ a partir de um conjunto de resoluções pré-definidas ($y$).
   **Research Question.** O que poderia ser considerado como *melhor resolução?*
   Leitura recomendada para o exercício: [46]

4. Crie uma rede convolucional que realize a classificação de imagens. Sua arquitetura pode conter **apenas** camadas convolucionais, *padding* e (opcionalmente) operações de

*reshape/flatten*. A arquitetura **não** deve conter camadas *fully-connected*.
Leitura recomendada para o exercício: [16]

5. Utilize a técnica de *loss landscape* para visualizar o espaço de parâmetros de uma rede neural (convolucional ou não) treinada. O código para gerar o *loss landscape* está disponível no github da disciplina.
Leitura recomendada para o exercício: [29]

# 7 Residual Networks

1. Conforme visto em aula, para facilitar a construção de redes complexas e profundas uma prática comum é desenvolver *building blocks* e, então, replicá-los para obter a arquitetura final. Proponha um *building block* com no mínimo 6 componentes. Feito isso, construa uma arquitetura de rede neural repetindo o *building block* proposto $n$ vezes.

2. Faça o mesmo que o exercício anterior, porém, agora insira *skip connections* **entre** *building block*.
Leitura recomendada para o exercício: [13]

3. A arquitetura abaixo apresenta o problema de *miss matching* que ocorre quando a operação de soma *point-wise* entre a entrada (*shortcut*) e saída da camada residual não possuem correspondência. O que deveria ser alterado para que a arquitetura funcionasse corretamente?

```
...
d = 2
inputs = keras.layers.Input(shape=X.shape[-1])
x = inputs
x = layers.Dense(d, activation='relu')(x)

for _ in range(n_blocks):
 shortcut = x
 x = layers.Dense(d, activation='relu')(x)
 x = layers.Dense(d * 2, activation='relu')(x)
 x = layers.Dense(d * 4, activation='relu')(x)
 x = layers.Dense(d * 2, activation='relu')(x)
 x = x + shortcut
 d = d * 2
 x = layers.Dense(d, activation='relu')(x)

outputs = layers.Dense(n_classes, activation='softmax')(x)
model = tf.keras.Model(inputs, outputs)
```

Código 3: Exercício 3

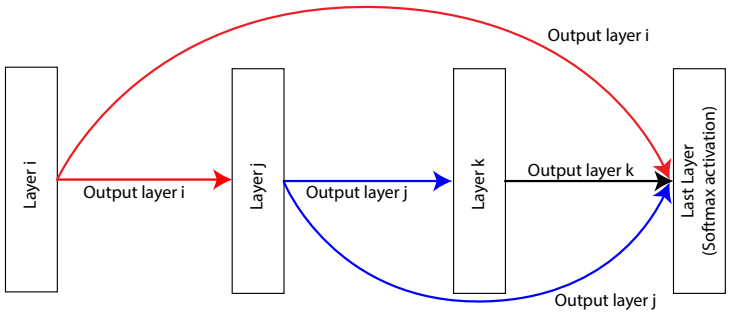


Figure 2: Exemplo de arquitetura para o Exercício 4. Formalmente, a entrada para a última camada $L$ é $\bigcup_{i=1}^{L-1} y_i$, onde $y_i$ corresponde à saída da $i$th camada.

4. Uma característica inerente de redes residuais é que elas possibilitam lidar com o *vanishing gradient* uma vez que o gradiente chega tanto em $f(x)$ quanto em $x$ (lembre-se que em redes residuais $y_i = f_i(x) + x$). Outra forma de lidar com o *vanishing gradient* é conectar diretamente todas as camadas ocultas à camada de saída (ex., camada *fully connected* com a ativação *Softmax*). Proponha uma arquitetura para lidar com o *vanishing gradient* agregando a representação das camadas por meio da operação de concatenação (ex.: ). A Figura 2 ilustra como seria tal arquitetura. Por simplicidade, você pode considerar uma rede MLP simples. Após elaborar sua arquitetura, pense em quais limitações a operação de concatenação mitiga em relação à operação de soma (*point-wise*) proposta inicialmente para as redes residuais.
   Leitura recomendada para o exercício: [19]

5. ($\star$) Uma característica interessante das redes residuais é que elas permitem que o fluxo de informações (representação) siga caminhos diferentes através da rede. Desta forma, as camadas podem depender fracamente umas das outras. De acordo com trabalhos da literatura, essa característica permite *apagar* algumas camadas durante o processo de *forward* sem comprometer a habilidade de predição da rede. Formalmente, considere $y_i$ a saída da $i$th camada de uma rede residual, $y_i = f_i(y_{i-1}, \theta_i) + y_{i-1}$. A partir desta descrição, durante o *forward* podemos zerar os pesos de $\theta_i$ levando à seguinte transformação $y_i = 0 + y_{i-1}$, que é equivalente a propagar somente a entrada $y_{i-1}$.

   **Research Question.** Como identificar quais camadas residuais não afetam a habilidade preditiva da rede. É possível desabilitar um conjunto de camadas para cada amostra $x$ sem alterar sua classificação? (Observação: lembre-se de retornar os valores de $\theta_i$ ao apagar uma nova camada $j$)
   Leitura recomendada para o exercício: [43, 4, 12]

# 8 Data Augmentation

1. Elabore um experimento para mensurar a importância de introduzir quantidade ou diversidade de amostras durante o treinamento de redes neurais.

2. Proponha uma técnica de aumento de dados fora do domínio de visão computacional. Verifique a efetividade da técnica e discuta os resultados.

3. ($\star$) Uma maneira de lidar com amostras adversariais é gerar e introduzir (*adversarial training*) tais amostras como parte dos exemplos de treinamento. Estudos relativamentre recentes, entretanto, vêm demonstrando resultados positivos ao lidar com cenários adversariais aplicando *data augmentation* nas amostras de teste (processo tipicamente denominado *testing time adaptation*). Por exemplo, Guo et al. [10] evidenciaram um fenômeno denominado *scaled prediction consistency*. A partir desse fenômeno, os autores sugerem aumentar o tamanho das imagens de entrada ($x$) e verificar uma pontuação de confiança (veja Equação 2 do artigo): se a confiança diminuir, a imagem corresponde a uma amostra benigna; caso contrário, a amostra é adversarial. Liu et al. [30] observaram que modelos infectados por ataques *backdoor* exibem uma robustez à corrupção semântica diferente para amostras com *trigger*. Os autores exploram essa observação para detectar amostras com *triggers*. A partir desses *insights*, proponha uma forma de *data augmentation* no teste para aumentar a robustez adversarial.
   Leitura recomendada para o exercício: [10, 14, 30]

# 9 Transformers e MLP Modernos

1. Apesar do sucesso dos transformers na automatização de diferentes tarefas cognitivas, seu custo computacional (principalmente nas fases de treinamento) é bastante elevado. No contexto de dados tabulares, TabPFN surge como uma alternativa eficiente. Infelizmente, TabPFN é limitado a um número de máximo de amostras de treinamento e *features*. Como contornar tais limitações? Formalize sua solução.
   Leitura recomendada para o exercício: [15]

2. Elabore um arquitetura MLP com skip-connection. A arquitetura deve ser construída utilizando a ideia de building blocks.

# 10 Transfer Learning e Knowledge Distillation

1. O código abaixo possui um erro semântico no contexto de *transfer learing*. Qual o erro e como podemos corrigi-lo?

```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.models import Model
```

```
from tensorflow.keras.layers import *

model = ResNet50(weights='imagenet')

#Target Dataset
n_samples, n_classes = 500, 10
X = np.random.rand(n_samples, 224, 224, 3)
y = np.random.randint(0, n_classes, size=n_samples)
y = np.eye(n_classes)[y]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

X_train_mean = np.mean(X_train, axis=0)
X_train -= X_train_mean
X_test -= X_train_mean

H = model.get_layer(index=-2).output
H = Flatten()(H)
H = Dense(n_classes, activation='softmax')(H)
model = Model(inputs=model.input, outputs=H)

model.compile(loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=1)
y_pred = model.predict(X_test)

acc = accuracy_score(np.argmax(y_test, axis=1), np.argmax(model
  .predict(X_test), axis=1))
```

Código 4: Código para Exercício 1

2. Suponha uma rede convolucional treinada sobre um vasto conjunto de dados (i.e., ImageNet – *source*) com imagens RGB de resolução $224 \times 224$. Suponha um conjunto de dados composto por imagens de resolução diferente que desejamos transferir o conhecimento dessa rede. Nesse cenário como podemos realizar *transfer learning*. Evite a solução óbvia de reescalar as imagens do *dataset target* para $224 \times 224$.

3. Implemente a técnica *linear probing* de *transfer learning*. Intuitivamente, os *datasets source* e *target* precisam ser diferentes.

4. Implemente a transferência de aprendizado seguindo os seguintes passos: (i) insira a *head* da tarefa *target*; (ii) congele os pesos do *backbone*; (iii) após algumas iterações de *fine-tuning* descongele o *backbone* para permitir que a rede inteira treine de maneira *end-to-end*.

Compare esse processo com um processo naive onde após inserir a *head target* toda a rede é treinada. Tente observar/argumentar se existe alguma vantagem em realizar o passo (ii) e (iii).
Leitura recomendada para o exercício: [17]

5. Implemente a técnica de *transfer learning* onde tanto o *backbone* quanto a parte de classificação são treinadas juntamente (isto é, *end-to-end*). Teste a solução para diferentes bases de dados *targets* e compare com a técnica *linear probing*. A partir dos resultados obtidos, o que podemos inferir sobre a representação do modelo (em particular, do *backbone*).
Leitura recomendada para o exercício: [7]

# 11 Parameter Efficient Fine-Tuning

1. ($\star$) Uma forma alterativa para ajustar os parâmetros de grandes modelos envolve a técnica *Low-Rank Adaptation* (LoRA) [18]. LoRA permite explorar o potencial de LLMs em tarefas especificas, adaptando os modelos a novos domínios por meio de *fine-tuning*.

   A técnica LoRA introduz um ajuste residual $\Delta W$ aos pesos pré-treinados do modelos, $W$, e parte da hipótese que $\Delta W$ é uma matriz *low-rank*. Em outras palavras, alterações ($\Delta W$) nos pesos $W$ exibem estrutura *low-rank*. Desta forma, LoRA reformula $\Delta W$ como o produto de duas matrizes $A \in \mathbb{R}^{r \times n}$ e $B \in \mathbb{R}^{m \times r}$, com $r \ll min(n, m)$, e permite reduzir o número de parâmetros treinados de $m \times n$ para $r \times (m + n)$. Utilizando LoRA, os pesos de uma camada $i$ tornam-se $W_i + \Delta W_i \approx W_i + B_i A_i$.

   Uma prática comum é utilizar o mesmo valor $r$ ao longo de todas as camadas do modelo, com os valores $r = 8$ e $r = 16$ sendo os mais populares [44]. Entretanto, estudos sugerem que algumas camadas apresentam comportamentos diferentes ao adotar valores distintos de $r$ [56]. Esforços nessa direção definem os valores de $r$ para cada módulo LoRA dinamicamente [52].
   **Research Questions.** Algumas camadas possuem preferência por valores de $r$ diferentes? Como podemos definir um $r$ adequado para cada camada?
   Leitura recomendada para o exercício: [52]

2. ($\star$) Alinhado com a questão anterior, estudos demonstram que algumas camadas, ao receberem LoRA, desempenham um papel mais importante na habilidade preditiva do modelo (veja, por exemplo, a Figura 1 do trabalho de Zhang et al. [56] e Tabela 5 do trabalho de Huang et al. [20]). Nessa direção, Zhou et al. [58] confirmou que o LoRA de algumas camadas possuem pouco impacto em tarefas específicas ao passo que outras camadas exibem efeitos mais significativos. A partir dessa observação, os autores propõem adicionar módulos LoRA treináveis apenas em algumas camadas e compartilham um mesmo módulo LoRA nas demais. De maneira geral, os trabalhos acima sugerem que nem todas as camadas precisam de LoRA. Em outras palavras, apenas algumas representações internas do modelo precisam ser atualizadas.
   **Research Question.** Como definir sistematicamente quais camadas devem receber módulos LoRA?
   Leitura recomendada para o exercício: [58, 55]

3. (⋆) O trabalho de Zhao et al. [57] propõe sintetizar o conhecimento de múltiplos LoRAs em um único e unificado LoRA por meio de concatenação. A ideia geral do método envolve decompor as colunas da matriz $A$ e as linhas correspondentes da matriz $B$ em unidades denominadas *Minimum Semantic Unit* (MSU). A partir desta reformulação mais fina de $A$ e $B$, os autores clusterizam os MSUs de diferentes LoRAs (várias matrizes $A$ e $B$ provindas de diferentes tarefas, ranks, etc.). Após o processo de clusterização, cada centroide $\mu_i$ contribui para um único *rank* em um LoRA unificado (*merged LoRA*). Formalmente, o novo módulo LoRA combinado resulta da construção de novas matrizes de projeção $A'$ e $B'$ a partir dos centróides:

$$A' = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_k \end{bmatrix}, \quad B' = \begin{bmatrix} b_1^T & b_2^T & \dots & b_k^T \end{bmatrix}, \tag{1}$$

onde cada centróide $\mu_i = [a_i, b_i]$ compõem um $a_i$ e $b_i$ (isto é, um *rank*).

Além dos resultados positivos, o trabalho de Zhao et al. [57] demonstra que os MSUs são invariantes a permutação e possuem a propriedade de equivalência nas operações de concatenação e soma.
**Research Question.** Como produzir centroides de qualidade? Como evitar conflito de conhecimento entre os MSUs? É possível identificar e descartar MSUs irrelevantes ou redundantes? É possível realizar a combinação dos MSUs de maneira *data-drive* (isto é, levando os dados em consideração)?
Leitura recomendada para o exercício: [57]

# 12 Model Merging

1. (⋆) O trabalho precursor de *model merging* propõe a utilização de um vetor de tarefa, $\tau$, que especifica uma direção no espaço de pesos do modelo pré-treinado ($\theta_{pre}$) [21]. Os autores argumentam que o movimento na direção $\tau$ melhora o desempenho do modelo na tarefa $tk$. A partir disso, Ilharco et al. [21] calculam $\tau$ como a diferença entre o modelo original e sua versão ajustada (*fine-tuned*) – $\theta_{ft}^{tk}$ – na tarefa $tk$. Mais concretamente, $\tau = \theta_{ft}^{tk} - \theta_{pre}$.
**Research Question.** Existem formas mais promissoras de produzir vetores de tarefas?

2. (⋆) *Model Merging* surge como uma técnica simples e promissora capaz de criar um único modelo que performe bem em diversas tarefas. Yang et al. [45] observou que o coeficiente $\lambda$ desempenha um papel importante para o sucesso da técnica *model merging* (veja, por exemplo, a Figura 1 de seu trabalho). Neste trabalho, os autores apontam que um dos desafios encontrados envolve determinar os coeficientes $\lambda$ que facilitam a integração ideal de múltiplas tarefas. O trabalho de Yoshida et al. [48] também aponta a importância de atribuir adequadamente os valores de $\lambda$. Em particular, aplicar $\lambda = 1$ compromete negativamente os resultados (veja, por exemplo, as Tabelas 1 e 2 de seu trabalho).

   Yang et al. [45] também sugerem que atribuir $\lambda$s diferentes para cada camada dos modelos levam a resultados positivos no processo de *model merging* (veja, por exemplo, o tópico

*Layer-wise AdaMerging* da Seção 3.2. do artigo).
**Research Questions.** Como definir, de maneira sistemática, um $\lambda$ adequado para cada modelo? Isto é, como ponderar a contribuição dos modelos ajustados $\theta_{ft}^{t_k}$ durante o processo de *merging*? Como definir um $\lambda$ para cada camada?
Leitura recomendada para o exercício: [45]

3. ($\star$) Stoica et al. [40] observou que modelos ajustados com LoRA exibem desempenho inferior no processo de *merging* comparado a modelos ajustados com *fine-tuning* tradicional. Os autores argumentam que isso ocorre devido a um desalinhamento dos modelos e propõem aplicar *Singular Value Decomposition* (SVD) para alinhar os pesos de modelos ajustados com LoRA no processo de *merging*.

   **Research Questions.** Todos os modelos deveriam ser considerados durante o processo de *merging*? (Note que isso corresponde a uma variação do problema anterior, onde $\alpha_{t_k} = 0$). Existem modelos (tarefas) que desempenham um papel mais importante? Como identificar modelos desalinhados?

# 13 Neural Phylogeny

1. Desenvolva um setup experimental para estudar o conceito de *neural phylogeny*. Para facilitar, crie modelos pequenos e empregue dados sintéticos.

2. ($\star$) Proponha um método para detectar *neural phylogeny*.
   Leitura recomendada para o exercício: [49, 50, 47]

# 14 GreenAI

1. ($\star$) Grandes modelos baseados em redes neurais influenciam diversos aspectos da sociedade moderna e impulsionam o surgimento de novas tecnologias [6]. O paradigma atual envolve treinar esses modelos em conjuntos extremamente grandes de dados (*web-scale*), capacitando-os a extrapolar sua habilidade preditiva para tarefas não observadas anteriormente. Por outro lado, o treinamento com dados em larga escala impõe diversos desafios como o alto custo computacional [34, 6]. Nessa direção, *data pruning* e *coresets* compreendem um tema importante para o treinamento de grande modelos de maneira mais eficiente, identificando e eliminando dados menos importantes para o aprendizado do modelo.
   **Research Questions.** Como definir quais dados remover (isto é, definir a qualidade do dado) antes ou durante a fase de treinamento dos modelos? Quais fatores possibilitam reduzir a quantidade de dados sem prejudicar a efetividade do treinamento? Após a aplicação de mecanismos de seleção de dados, os modelos treinados tornam-se sensíveis ou robustos a ataques adversariais?
   Leitura recomendada para o exercício: [37, 38, 51]

2. ($\star$) A literatura moderna sobre seleção de dados confirma que técnicas sofisticadas e computacionalmente custosas superam marginalmente heurísticas aleatórias [37, 38, 2]. Por exemplo, Okanovic et al. [37] evidenciam que a amostragem aleatória, repetida ao longo

do treinamento, constitui um método simples e eficaz para a seleção de dados, superando técnicas mais complexas. Similarmente, Qui et al. [38] propõem eliminar dados de maneira aleatória (baseado na distribuição da *loss*) e reescalar os gradientes das amostras restantes.
**Research Questions.** Quais aspectos levam heurísticas simples (ex. uma seleção aleatória) a alcançarem resultados competitivos? Durante a trajetória do aprendizado, existem períodos mais adequados para realizar uma seleção de dados aleatória?
Leitura recomendada para o exercício: [37, 38, 51]

# References

[1] Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in CNNs. In *International Conference on Learning Representations (ICLR)*, 2024.

[2] Hoyong Choi, Nohyun Ki, and Hye Won Chung. BWS: best window selection based on sample scores for data pruning across broad ranges. In *International Conference on Machine Learning (ICML)*, 2024.

[3] Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. The efficiency misnomer. In *International Conference on Learning Representations (ICLR)*, 2022.

[4] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning (ICML)*, 2021.

[5] Lyndon R. Duong, Jingyang Zhou, Josue Nassar, Jules Berman, Jeroen Olieslagers, and Alex H. Williams. Representational dissimilarity metric spaces for stochastic neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

[6] Yoshua Bengio et al. International AI safety report. 2025.

[7] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2Toe: Utilizing intermediate representations for better transfer learning. In *International Conference on Machine Learning (ICML)*, 2022.

[8] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. In *International Conference on Learning Representations (ICLR)*, 2021.

[9] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[10] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *International Conference on Learning Representations (ICLR)*, 2023.

[11] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubik's cube: Twisting resolution, depth and width for tinynets. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Zhiyuan He, Yijun Yang, Pin-Yu Chen, Qiang Xu, and Tsung-Yi Ho. Be your own neighborhood: Detecting adversarial examples by the neighborhood relations built on self-supervised learning. In *International Conference on Machine Learning (ICML)*, 2024.

[15] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *International Conference on Computer Vision (ICCV)*, 2019.

[17] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Association for Computational Linguistics (ACL)*, 2018.

[18] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient hadamard high-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[21] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023.

[22] Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *International Conference on Learning Representations (ICLR)*, 2025.

[23] Max Klabunde, Tassilo Wald, Tobias Schumacher, Klaus Maier-Hein, Markus Strohmaier, and Florian Lemmerich. Resi: A comprehensive benchmark for representational similarity measures. In *International Conference on Learning Representations (ICLR)*, 2025.

[24] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods for multisensory integration in deep networks. In *Computer Vision and Pattern Recognition (CVPR*, 2023.

[25] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods emerge even in deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2024.

[26] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019.

[27] Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. AM-LFS: automl for loss function search. In *International Conference on Computer Vision (ICCV)*, 2019.

[28] Hao Li, Tianwen Fu, Jifeng Dai, Hongsheng Li, Gao Huang, and Xizhou Zhu. Autoloss-zero: Searching loss functions from scratch for generic tasks. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[29] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[30] Xiaogeng Liu, Minghui Li, Haoyu Wang, Shengshan Hu, Dengpan Ye, Hai Jin, Libing Wu, and Chaowei Xiao. Detecting backdoors during the inference stage based on corruption robustness consistency. In *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[31] Sepideh Mahabadi and Stojan Trajanovski. Core-sets for fair and diverse data summarization. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[32] Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J. Zico Kolter, and Chiyuan Zhang. In *International Conference on Machine Learning (ICML)*, 2023.

[33] Wojciech Masarczyk, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Miłoś, and Tomasz Trzcinski. The tunnel effect: Building data representations in deep neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[34] Jacob Morrison, Clara Na, Jared Fernandez, Tim Dettmers, Emma Strubell, and Jesse Dodge. Holistically evaluating the environmental impact of creating language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[35] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations (ICLR)*, 2021.

[36] Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations. *Transactions on Machine Learning Research*, 2022.

[37] Patrik Okanovic, Roger Waleffe, Vasilis Mageirakos, Konstantinos E. Nikolakakis, Amin Karbasi, Dionysios S. Kalogerias, Nezihe Merve Gürel, and Theodoros Rekatsinas. Repeated random sampling for minimizing the time-to-accuracy of learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[38] Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, and Yang You. Infobatch: Lossless training speed up by unbiased dynamic data pruning. In *International Conference on Learning Representations (ICLR)*, 2024.

[39] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *International Conference on Learning Representations (ICLR)*, 2018.

[40] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with SVD to tie the knots. In *International Conference on Learning Representations (ICLR)*, 2025.

[41] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[42] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[43] Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[44] Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations (ICLR)*, 2025.

[45] Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, 2024.

[46] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] Nicolas Yax, Pierre-Yves Oudeyer, and Stefano Palminteri. PhyloLM: Inferring the phylogeny of large language models and predicting their performances in benchmarks. In *International Conference on Learning Representations (ICLR)*, 2025.

[48] Kotaro Yoshida, Yuji Naraki, Takafumi Horie, Ryosuke Yamaki, Ryotaro Shimizu, Yuki Saito, Julian McAuley, and Hiroki Naganuma. Mastering task arithmetic: $\tau$jp as a key indicator for weight disentanglement. In *International Conference on Learning Representations (ICLR)*, 2025.

[49] Runpeng Yu and Xinchao Wang. Neural lineage. In *Computer Vision and Pattern Recognition (CVPR)*, June 2024.

[50] Runpeng Yu and Xinchao Wang. Neural phylogeny: Fine-tuning relationship detection among neural networks. In *International Conference on Learning Representations (ICLR)*, 2025.

[51] Suqin Yuan, Runqi Lin, Lei Feng, Bo Han, and Tongliang Liu. Instance-dependent early stopping. In *International Conference on Learning Representations (ICLR)*, 2025.

[52] Emanuele Zangrando, Sara Venturini, Francesco Rinaldi, and Francesco Tudisco. dEBORA: Efficient bilevel optimization-based low-rank adaptation. In *International Conference on Learning Representations (ICLR)*, 2025.

[53] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[54] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 2022.

[55] Jun Zhang, Jue WANG, Huan Li, Lidan Shou, Ke Chen, Yang You, Guiming Xie, Xuejian Gong, and Kunlong Zhou. Train small, infer large: Memory-efficient loRA training for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[56] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.

[57] Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, and Fei Wu. Merging loRAs like playing LEGO: Pushing the modularity of loRA to extremes through rank-wise clustering. In *International Conference on Learning Representations (ICLR)*, 2025.

[58] Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. Lora-drop: Efficient lora parameter pruning based on output evaluation. In *International Conference on Computational Linguistics (COLING)*, 2025.