

# Finding interpretable class-specific patterns through efficient neural search

Seminário 2 – Aprendizado Descritivo  
2025.1

# Historiador

Caio Jorge Carvalho Lara

Juan Marcos Braga Faria

Luisa Vasconcelos de Castro Toledo

Luiza Sodré Salgado

Mateus Reis Evangelista

Samuel Henrique Miranda Alves

# Contextualização

- **Descoberta de subgrupos (Subgroup Discovery)** é uma tarefa de **aprendizado descritivo supervisionado** cujo objetivo é encontrar **padrões locais** (ou regras) que associam subconjuntos dos dados a uma **classe-alvo** ou uma **variável de interesse**, de forma **estatisticamente significativa e interpretável**.
- Um **subgrupo** é geralmente expresso como uma **conjunção de condições (regra)** sobre os atributos, por exemplo:

IF (idade > 60)  $\wedge$  (nível de glicose > 140) THEN classe = “diabetes”

- O objetivo é que esse subgrupo **destaque uma relação interessante entre atributos e a classe**, revelando padrões que possam ser usados para entender os dados, formular hipóteses ou apoiar decisões em domínios como medicina, marketing ou ciências sociais.
- Ex: Encontrar padrões de genes que diferenciem câncer de mama de um tecido saudável.

# Motivação e Justificativa

Cerne da Motivação: **entender melhor os mecanismos que distinguem diferentes classes de dados.**

Foco em domínios de alta complexidade como a **biologia molecular**, onde os dados são **extremamente dimensionais** (com centenas de milhares de atributos).

Abordagens **preditivas** priorizam desempenho, mas são opacas.

Contexto científico e biomédico exige foco na **interpretação** → **Por que** uma amostra pertence a uma classe? **Quais padrões distinguem as classes.**

## Justificativas:

- **Falta de escalabilidade para milhares de atributos**
- **Pouca robustez a ruído**
- **Dificuldade de interpretação em redes neurais convencionais (caixa-preta)**

# Trabalhos Relacionados

**CART:** Baseado em árvores de decisão, é um método clássico que procura encontrar todas as circunstâncias em que uma classe assume uma distribuição excepcional, gerando vários resultados, que são altamente redundantes e espúrios.

**SPUMANTE:** Sofre de explosão de padrões. Mesmo com dados pequenos, eles frequentemente encontram dezenas de milhares de padrões redundantes, devido à falta de correção de múltiplos testes de hipóteses.

**PREMISE:** Busca um Conjunto não redundante de padrões específicos de classe que, juntos, descrevem bem os dados. Funciona bem com dados pequenos, mas, como se baseiam em heurísticas de busca combinatória que são (pelo menos) quadráticas em termos de número de características, são, em sua maioria, inaplicáveis a dados de alta dimensão.

**CLASSY:** Foca em previsão em vez da descrição e, portanto, perde detalhes importantes. Além disso, se baseia em otimização combinatória, o que impede de escalar para conjuntos de dados de alta dimensão.

**RLL:** Se concentra na precisão da classificação em vez da descoberta completa de regras. Já o DIFFNAPS combina reconstrução de dados com classificação para descobrir as explicações interpretáveis por humanos relevantes para as classes.

# Comparação com Métodos Clássicos

Critério	SD-Map / Beam Search / SSDP	DIFFNAPS (artigo)
Tipo de modelo	Simbólico (regras)	Neuro-simbólico (rede neural binária + regras implícitas)
Forma de saída	Regras do tipo IF-THEN	Representações binárias interpretáveis
Busca de padrões	Exploratória (DFS, beam search)	Otimização contínua com backpropagation
Escalabilidade (nº atributos)	Limitada (milhares, no máximo)	Alta (centenas de milhares de atributos)
Robustez a ruído	Moderada	Alta (testado empiricamente no artigo)
Tipo de supervisão	Métricas heurísticas por padrão	Loss supervisionado (classificação + reconstrução)
Diversidade de subgrupos	Implementada via heurísticas (ex: SSDP)	Emergem pela estrutura da rede + regularização
Domínios aplicáveis	Pequenos a médios; dados tabulares	Alta dimensionalidade, como expressão gênica

# Formalizador

Alexis Duarte Guimarães Mariz

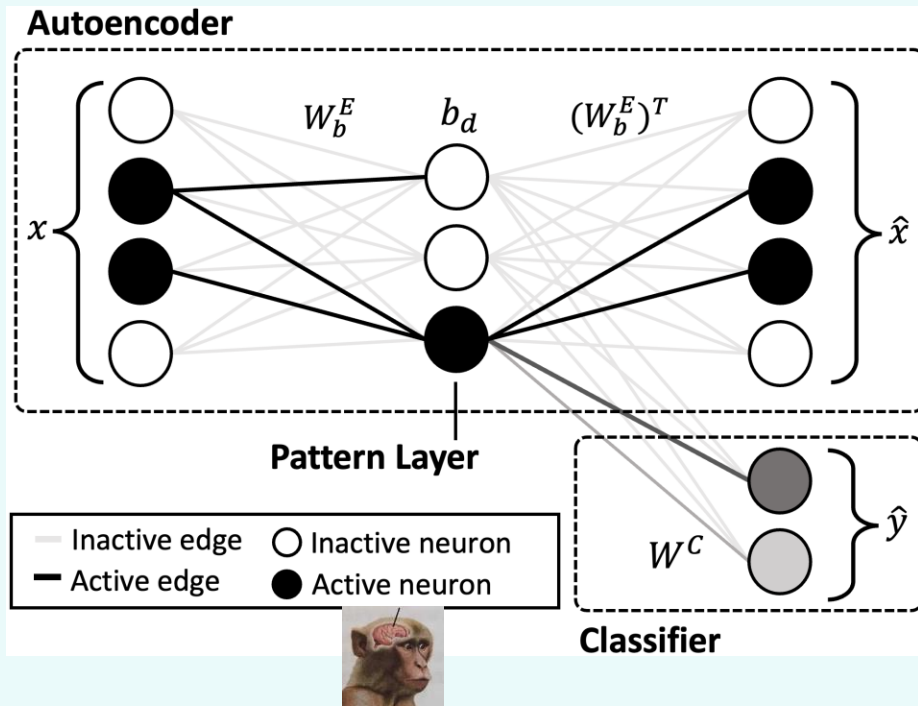
Amanda Mendes Pinho

João Vítor Fernandes Dias

**Kael Soares Augusto**

Lucas Xavier Veneroso

# DiffNaps in a Nutshell



Continuous		Discrete
$W^E = \begin{bmatrix} 0.02 & 0.87 & 0.02 & 0.87 \\ 0.2 & 0.01 & 0.04 & 0.9 \\ 0.0 & 0.99 & 0.8 & 0.09 \end{bmatrix}$	$\xrightarrow{\mathcal{B}(\cdot)}$	$W_b^E = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$
$b^T = [-2.3 \quad -1.4 \quad -1.1]$	$\xrightarrow{[\cdot]}$	$b_d^T = [-2 \quad -1 \quad -1]$
$W^C = \begin{bmatrix} 0.07 & 0.01 & 0.9 \\ 0.02 & 0.9 & 0.2 \end{bmatrix}$	$\xrightarrow{\mathcal{B}_{\tau_C}(\cdot)}$	$W_b^C = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
<b>Differential patterns:</b> $P^1 = \{\{2,3\}\}$ resp. $P^2 = \{\{1,4\}\}$		

- AutoEncoder Binarizado
- Classificador linear completamente conectado



# DiffNaps in Detail: Jumpscore

$$l_e(x, \hat{x}) = \sum_{j=1}^m ((1 - x_j)\alpha + x_j(1 - \alpha)) |x_j - \hat{x}_j|$$

A prova é trivial e  
fica como exercício  
para o leitor.

$$l_e(x, \hat{x}) = \sum_{j=1}^m ((1 - x_j)\alpha + x_j(1 - \alpha)) |x_j - \hat{x}_j|$$

$$\mathcal{L}(X, Y; \theta) = \sum_{i=1}^n \frac{f_D(x_i)}{n_k}$$

$$= \sum_{k=1}^K \frac{\lambda E(f_{W^E}(x_i))}{\mathbb{P}(k|p)} + r_b(\theta)$$

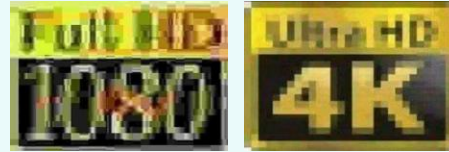
$$r_s(W) = \sum_{i \in W} \left( \sum_{j=1}^h W_{i,j} \right)^2$$

$$= \lambda_D(f_{W^D}(z))$$

$$r_b(W) = \sum_{w \in W} \min \{r(w), r(w-1)\}$$

$$r(w) = \kappa \|w\|_1 + \lambda \|w\|_2^2$$

# DiffNaps in Detail



Formato de entrada:

- Matriz binária de Samples x Features e vetor de Classes  $\in \{1, \dots, K\}$

Objetivo:

- Encontrar **padrões diferenciais** para uma **classe k**.

$$\mathbb{P}(p \mid k) = \frac{supp_k(p)}{n_k}$$

e

$$\mathbb{P}(k \mid p) = \frac{supp_k(p)}{supp(p)}$$

Um padrão  $p$  é tal que todos elementos de  $p$  se associam a features presentes.

- Queremos encontrar um " $p_k$ " onde
- a)  $Sup(X^k) > Sup(X \setminus X^k)$
- b) Classe  $K$  é a mais provável de ter esse padrão

O resultado final são conjuntos  $P^k$  para todo  $k$  com os padrões  $p_k$

# Forward pass



LambdaE é centrado no learned bias

$$\lambda_E$$

$$z = f_E(x) = \lambda_E(f_{W_b^E}(x))$$

$$\hat{x} = f_D(z) = \lambda_D(f_{W_b^D}(z))$$

# Objective function

Alpha é uma medida de esparcidade da matriz de Samples x Features:  $\#1/(\#1 + \#0)$

$$l_e(x, \hat{x}) = \sum_{j=1}^m ((1 - x_j)\alpha + x_j(1 - \alpha)) |x_j - \hat{x}_j|$$

Pegamos o "ground truth"  $Y_k$  e o resultado  $\hat{Y}_k$  do classificador usando "cross entropy loss":

$$l_c(y, \hat{y}) = \sum_{k=1}^K y_k \log(\hat{y}_k)$$

Penalizamos pesos excessivamente grandes (não sucintos)

$$r_s(W) = \sum_{i=1}^m \left( \sum_{j=1}^h W_{i,j} \right)^2$$

W-shaped regularizer

$$r_b(W) = \sum_{w \in W} \min \{r(w), r(w - 1)\}$$

$$r(w) = \kappa \|w\|_1 + \lambda \|w\|_2^2$$

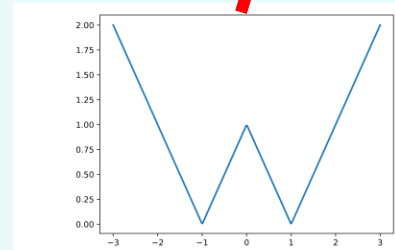
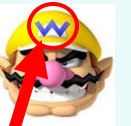
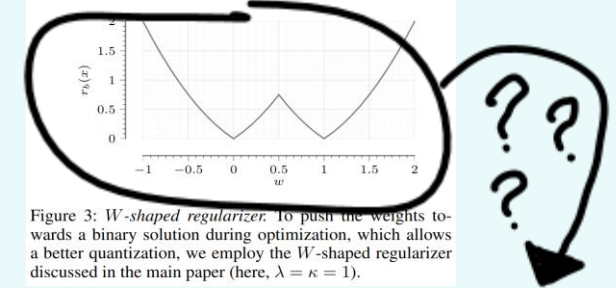


Figure 2: **W-shaped** regularizer for binary quantization.

# Objective function

$$l_e(x, \hat{x}) = \sum_{j=1}^m ((1 - x_j)\alpha + x_j(1 - \alpha)) |x_j - \hat{x}_j|$$

$$l_c(y, \hat{y}) = \sum_{k=1}^K y_k \log(\hat{y}_k)$$

$$\mathcal{L}(X, Y; \theta) = \sum_{i=1}^n l_e(y_i, \hat{y}_i) + \lambda_c l_c(x_i, \hat{x}_i) + r_s(W^E) + r_b(\theta)$$

**F**( $\frac{+}{\parallel} \frac{\pm}{\perp}$ )

$$r_s(W) = \sum_{i=1}^m \left( \sum_{j=1}^h W_{i,j} \right)^2 \quad r_b(W) = \sum_{w \in W} \min \{r(w), r(w-1)\}$$

$$r(w) = \kappa \|w\|_1 + \lambda \|w\|_2^2$$

# Back Propagation – Simplified:

Usamos a derivada para fazer o back propagation por meio de um "Straight Through Estimator (STE)", que é feito para redes binarizadas como a nossa.

$$\frac{\partial f_{W_b^E}}{\partial W^E} := g_u x^\top \qquad \frac{\partial f_{W_b^E}}{dx} := (W^E)^\top g_u$$

Para não penalizar neurônios desligados (o que poderia gerar neurônios negativos), apenas ajustamos neurônios ativados.

$$\frac{\partial \lambda_E}{\partial b} := \begin{cases} g_u & \text{se } \lambda_E(x) = 1 \\ 0 & \text{se } \lambda_E(x) = 0 \end{cases}$$
$$\frac{\partial \lambda_E}{\partial x} := \begin{cases} g_u & \text{se } \lambda_E(x) = 1 \\ \max(0, g_u) & \text{se } \lambda_E(x) = 0 \end{cases}$$

# Metodologista

Victor Gabriel Moura Oliveira (Apresentador)

Henrique Rotsen

Alexandre Cassimiro Silva Araújo

Wesley Marques Daniel Chaves

Caíque Bruno Fortunato

Arthur Yochio R. Codama

# Estratégia experimental: baseline

## Técnicas usadas como baseline:

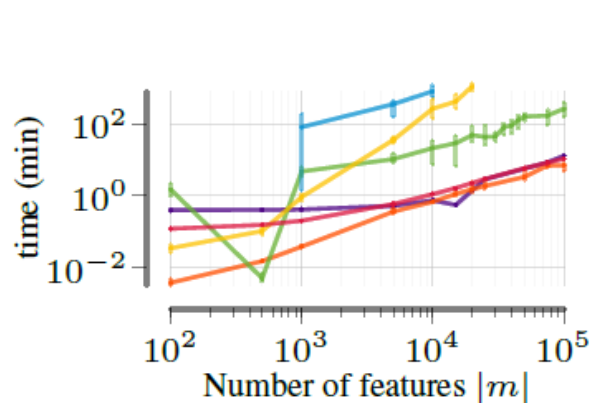
- Decision Trees (CART, Breiman 1984):
  - Algoritmo clássico de aprendizado supervisionado.
- Significant Pattern Mining (SPUMANTE, Pellegrina et al. 2019):
  - Foca na extração de padrões estatisticamente significativos. (classes binárias)
- MDL-based Label-Descriptive (PREMISE, Hedderich et al. 2022):
  - Utiliza o princípio da menor descrição (Minimum Description Length) para selecionar subconjuntos de padrões que melhor explicam os rótulos dos dados. (classes binárias)
- Classification Rule Learning (CLASSY, Proença & van Leeuwen 2020):
  - Aprende regras de classificação interpretáveis, com foco em precisão e simplicidade, combinando heurísticas com estratégias de busca.
- Neuro-symbolic Rule Learning (RLL, Wang et al. 2021):
  - Integra aprendizado simbólico e redes neurais para induzir regras de classificação interpretáveis a partir de representações apreendidas.



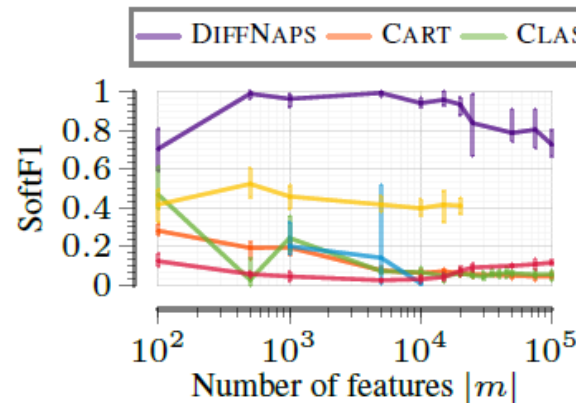
# Estratégia experimental: dados sintéticos

## Geração:

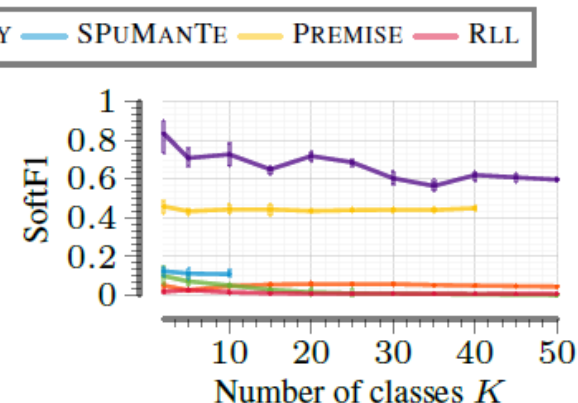
- 1) Amostramos 10 padrões específicos para cada classe e 20 padrões a serem compartilhados.
- 2) Em cada transação, introduzimos 2 padrões comuns e 3 padrões específicos para a classe da transação.
- 3) Construímos classes  $k$  de tal forma que  $\mathbb{P}(k \mid \tilde{p} \in P_k) = 0.9$ , ou seja, dado que  $\mathbb{P}(k \mid p) = \frac{\text{supp}_k(p)}{\text{supp}(p)}$  devemos ter que 90% do suporte para os padrões específicos da classe  $k$  deve se referir a transações rotuladas como  $k$ .
- 4) Adicionamos ruído aleatório ("flipar" 0s e 1s)



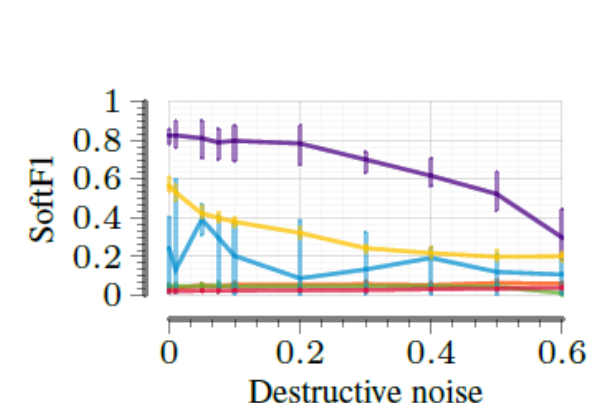
(a) Runtime (lower is better)



(b) F1-score (higher is better)



(c) F1-score (higher is better)



(d) F1-score (higher is better)

# Estratégia experimental: dados reais

- Cardio:
  - Dados fenotípicos cardíacos (Ulianova 2017)
- Disease
  - Diagnóstico de doenças (Patil e Rathod 2020)
- BRCA-N e BRCA-S
  - Expressão gênica para câncer de mama (TCGA)
- Genomes
  - Variação genética humana (Fischer and Vreeken 2020)

# Análise quantitativa

**Resultados Quantitativos:** Como o ground truth não é conhecido, os autores avaliaram:

- Quantidade de padrões extraídos.
- Comprimento médio dos padrões.
- Área sob a curva (AUC) da porcentagem de dados cobertos pelos padrões, ordenados pela probabilidade de pertencer a uma classe dado o padrão.
  - Mede o quanto os padrões cobrem dos dados e quão específicos são para a classe.
  - Quanto maior a área melhor, pois indica padrões que cobrem mais dados da classe e são mais específicos.

**Observações:**

- DiffNaps vai bem em todos os datasets, obtendo sempre o maior ou segundo maior AUC
- Premise e SPuManTe não escalam bem para bases com alta dimensionalidade.
- RLL só retornou padrões no dataset Cardio, e, por isso não foi considerado.

Dataset	$n$	$m$	$K$	DIFFNAPS (ours)			CART			CLASSY			PREMISE			SPUMANTE		
				$\#P$	$ \overline{P} $	AUC	$\#P$	$ \overline{P} $	AUC	$\#P$	$ \overline{P} $	AUC	$\#P$	$ \overline{P} $	AUC	$\#P$	$ \overline{P} $	AUC
Cardio	68k	45	2	14	2	.56	7k	6	<b>.62</b>	10	2	.36	28	1	.51	346	4	.56
Disease	5k	131	41	838	2	<b>.84</b>	1	2	.00	25	2	.11	187	3	<b>.84</b>	2k	3	.39
BRCA-N	222	20k	2	146	9	.91	1	2	.00	3	1	.45	–	–	–	4k	3	<b>.95</b>
BRCA-S	187	20k	4	1k	2	<b>.86</b>	22	2	.31	2	1	.23	–	–	–	0	0	0
Genomes	2.5k	225k	6	732	7	<b>.77</b>	127	4	.46	7	2	.36	–	–	–	–	–	–

# Análise Qualitativa

## **Differentiating Breast Cancer and Healthy Tissue**

- DIFFNAPS encontrou 146 padrões diferenciais de co-expressão gênica.
- Padrões refletem vias moleculares conhecidas para câncer (MAPK, WNT) e tecido saudável (lipólise, PPAR).
- Indica que os padrões capturam funções biológicas complexas relacionadas ao câncer.

## **Differentiating Cancer Subtypes**

- Padrões específicos foram identificados para subtipos como Luminal A e Luminal B.
- Luminal A: padrões relacionados a efeitos colaterais de tratamento (cardiomiopatia).
- Luminal B: padrões ligados ao metabolismo de esfingolípídios, importantes para sobrevivência celular.
- Revela diferenças moleculares e potenciais novos alvos terapêuticos.

## **Promising Novel Patterns**

- Padrões altamente específicos para classes (log-odds elevados).
- Alguns padrões não anotados em vias conhecidas, mas fortemente associados ao câncer.
- Indicam possibilidade de novas descobertas e tratamentos futuros.

# Assessor Social

Grupo: **ENILDA ALVES COELHO**

FÁBIO CÉSAR MARRA FILHO

GABRIEL TONIONI DUARTE

IASMIN CORREA ARAUJO

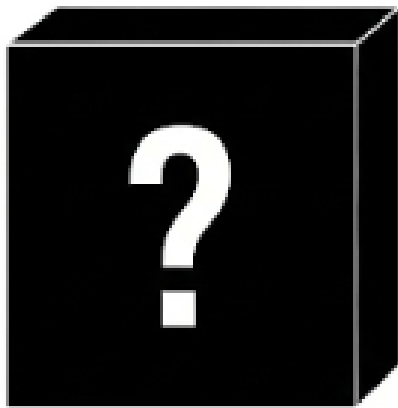
JOSE VINICIUS DE LIMA MASSARICO

LARISSA DUARTE SANTANA

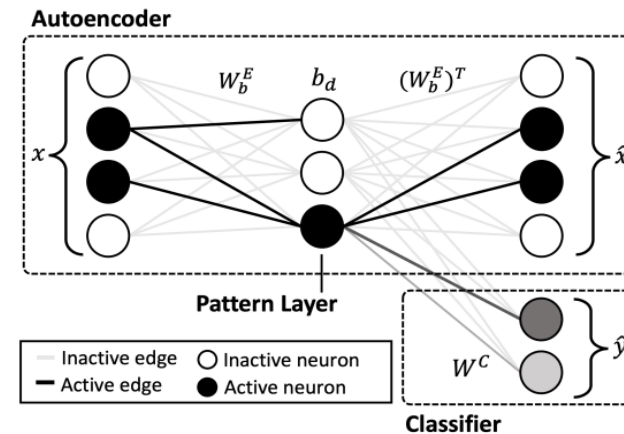
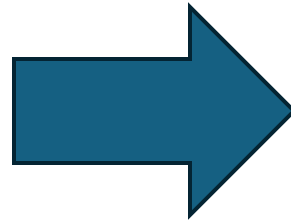
MARCELO LOMMEZ RODRIGUES DE JESUS

# Inovação tecnológica

- O modelo DIFFNAPS (supervisionado) foi projetado para que seus neurônios internos representem **padrões simbólicos e legíveis por humanos, sendo um modelo eficiente para altíssimas dimensões**



Ferramenta de automação (tomada de decisão)



Ferramenta de tomada de decisão informada  
Ferramenta de apoio à descoberta científica

## TRANSPARÊNCIA + EXPLICABILIDADE

# Cenários de uso

## Câncer Genome Atlas (TCGA data)

Diferenciação de tecidos cancerígenos e saudáveis

Diferenciação de subtipos de câncer

Padrões **interpretáveis**

De **alta qualidade**

capturam **classes** relacionadas a processos biológicos complexos

## Genome Dataset (CENÁRIO IDEAL)

Descobrir potenciais predisposições genéticas de indivíduos à doenças  
>> **avanços na detecção e tratamento precoce**



- **95% de chance de cura do câncer de mama** se diagnosticado em fase inicial (American Cancer Society)
- **Estima-se 18 mil mortes** anualmente devido ao câncer de mama no Brasil (INCA).

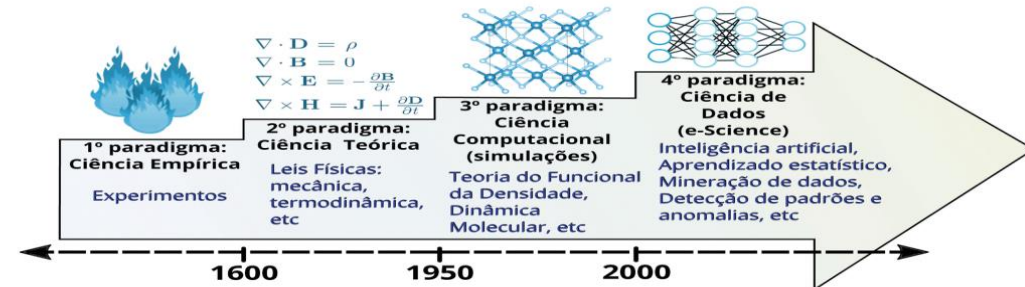
# Cenários de uso

- **Medicina de Precisão**

- Identificar padrões de risco para diversas doenças a partir dos dados
- padrões de comportamento nas mídias sociais >> depressão, ansiedade ou suicídio
- Padrões específicos podem ser associados a tratamentos personalizados a cada paciente
- Padrões podem contribuir para identificação de fármacos mais eficientes

- **Aceleração de descobertas científicas**

- Interpretabilidade dos padrões facilita a formulação de hipóteses e o raciocínio sobre fenômenos complexos, acelerando a pesquisa e a descoberta científica.
- O tempo médio desde a descoberta científica é de 10 a 15 anos, com alto custo



Fonte: SHLEDER & FAZZIO (2021) CC BY 3.0

- **Novas descobertas em diversas áreas**, além da medicina, física, biologia, entre outras.

- Descoberta de novos materiais na física e na química (SHLEDER & FAZZIO (2021))
- Padrões que levam a previsão de defeitos ou falhas de equipamentos (dados de sensores IoT)



# Potenciais riscos e danos à sociedade



Privacy break

•**Risco à Privacidade e Reidentificação de Indivíduos:** A capacidade de extrair padrões interpretáveis de dados de alta dimensão pode, em teoria, permitir a reidentificação de indivíduos. Essa informação pode ser usada indevidamente e levar à discriminação, exclusão social e prejuízos financeiros. Ex. \$\$\$ seguros saúde, etc.



Biased AI

•**Reforço de Vieses e Discriminação:** dados que contêm vieses históricos ou sociais podem aprender e replicar esses padrões discriminatórios. Tomadas de decisões "racionais", na verdade, perpetuam ou amplificam desigualdades existentes na história. Padrões explicativos, embora valiosos, podem ser mal interpretados por profissionais (como médicos, juízes ou gestores), levando a decisões equivocadas.



Inequality

•**Desigualdades:** diagnósticos mais precisos e personalizados podem abrir caminho para novos tratamentos, mas também aprofundar as desigualdades sociais e de saúde existentes, se acessível a poucos.



Data limitations

•**Limitação dos dados que representam a realidade:** modelos constituem uma simplificação da realidade, que muitas vezes é mais complexa e não está explicitamente representada pelos dados de entrada e não poderão ser capturados integralmente por um conjunto discreto de padrões. Isso significa que, apesar da interpretabilidade, há nuances e fenômenos que podem não ser totalmente compreendidos ou modelados.

**Alguns problemas são resolvidos e novos problemas surgem das inovações...**

**Como minimizar esses problemas? Esse debate deve ultrapassar as fronteiras da universidade....**

# Hacker

Grupo: DANIEL SCHLICKMANN BASTOS  
GABRIEL CASTELO BRANCO ROCHA  
GUILHERME BUXBAUM MARINHO GUERRA  
JOSE EDUARDO DUARTE MASSUCATO

LEONARDO CAETANO GOMIDE  
LUCAS MESQUITA ANDRADE  
VINICIUS LEITE CENSI FARIA

- O artigo por si só não mostra onde acessar o código
- Código Disponível em <https://github.com/nilspwalter/diffnaps>, em que houve uma verificação dos autores. Então basta Clonar e rodar de acordo com as instruções do repositório.

**diffnaps** Public

Watch 1 Fork 0 Starred 3

main 1 Branch 0 Tags

Go to file Add file Code

About

Finding Interpretable Class-Specific Patterns through Efficient Neural Search (AAAI '24)

Readme GPL-3.0 license Activity 3 stars 1 watching 0 forks Report repository

Releases

No releases published

Packages

No packages published

Languages

Python 100.0%

**Finding Interpretable Class-Specific Patterns through Efficient Neural Search**

**Autoencoder**

Continuous Discrete

$W^e = \begin{bmatrix} 0.02 & 0.87 & 0.02 & 0.87 \\ 0.2 & 0.01 & 0.04 & 0.9 \\ 0.0 & 0.99 & 0.8 & 0.09 \end{bmatrix} \xrightarrow{\mathcal{B}(\cdot)} W_b^e = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$

$b^T = [-2.3 \quad -1.4 \quad -1.1] \xrightarrow{[\cdot]} b_d^T = [-2 \quad -1 \quad -1]$

$W^c = \begin{bmatrix} 0.07 & 0.01 & 0.9 \\ 0.02 & 0.9 & 0.2 \end{bmatrix} \xrightarrow{\mathcal{B}_{rc}(\cdot)} W_b^c = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

**Pattern Layer**

**Classifier**

Differential patterns:  $P^1 = \{\{2,3\}\}$  resp.  $P^2 = \{\{1,4\}\}$

Legend: Inactive edge (dashed line), Active edge (solid line), Inactive neuron (open circle), Active neuron (filled circle)

Roteiro para recriar o artigo:

1. Clonar o Repositório
2. Baixar Dependências
3. Atualizar a base de Dados dentro da arquitetura
4. Rodar os Arquivos Python

# Como rodar

- Estas são as instruções no github. Eles tem todos os métodos implementados e basta fazer uma chamada do arquivo Python de cada experimento, se fossem dados reais seria necessário inserir o data set na arquitetura, dentro da pasta DATA

## 2. Folder organization

- **code:** Contains the code of our method, *Diffnaps*, and scripts to reproduce all the results presented in the paper
- **data:** Data used for the experiments for real data. Due to the upload limit *genomes* is not included
- **results:** Directory to store the results outputted by the scripts in **code**
- **appendix.pdf:** Contains appendix for the main paper

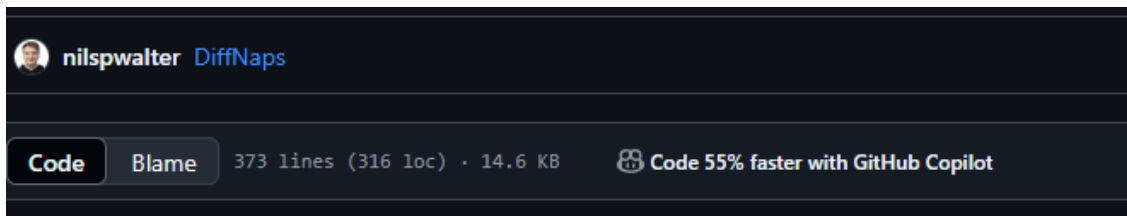
To run the experiments on **synthetic data**, the scripts `exp1.py` , `exp2.py` , `exp3.py` and `exp4.py` need to be executed. The experiments for scalability in number of features and number of classes can be found in `exp1.py` and `exp2.py` , respectively. The third experiment i.e. robustness to noise is split into two scripts. The file `exp3.py` contains the code for the additive noise and file `exp4.py` the code for the destructive noise.

To reproduce the results on **real data** the python script `real_exp.py` can be used. The dataset name corresponds to the names in the paper. So for  $\text{DATASET} \in [\text{cardio}, \text{disease}, \text{brca-n}, \text{brca-s}, \text{genomes}]$ ,

```
python3 real_exp.py -d DATASET ,
```

# Primeiros Problemas

- Eles pedem não só uma GPU pra rodar, como para rodar com dados reais esparsos eles pedem 40gb.
- A base de dados era Grande demais pro GitHub, eles tem um markdown com as instruções - é um código extenso em R de uma base mais leve usada tem mais de 2TB (BAM Files da TCGA (BRCA)), não foi reproduzi por falta de recursos. Seria necessário baixar tudo para fazer ao menos uma parcela dos dados



- Solução: Rodar no Collab com dados sintéticos. Eles se referem a cada uma das fases de teste citadas no artigo

# Dependências

## O que era sugerido

### 1. Required packages

- PyTorch
- Numpy
- Pandas

## O que acabou sendo necessário

- Numpy
- Scipy
- Pandas
- Matplotlib
- Seaborn
- Scikit –Learn
- Torch

# Segundo Problema: Versão do Python

O jeito que os dados Sintéticos era gerados usava a função `set()` com dicionários, incompatível com a versão atual do Python.

Solução 1: Trocar a versão do Python para o Python 3.9.18. Funciona quando rodar local (Pyenv) mas não é possível trocar a versão do Python do Collab

Solução 2: No arquivo `gen_synth_datasets.py`

Trocar a linha 12 de  
`return [sorted(random.sample(set(items),random.choice(range(*ran))))for i in range(n)]`

Por

`return [sorted(random.sample(list(items), random.choice(range(*ran)))) for i in range(n)]`

Ou seja, trocar o `set()` pelo `list()`

```
[100, 500, 1000, 5000, 10000, 15000, 20000, 25000, 50000, 100000]
Traceback (most recent call last):
  File "/content/diffnaps/code/exp1.py", line 78, in <module>
    exp_dict = experiment1([ 100, 500, 1000, 5000, 10000, 15000,20000,25000,50000,100000],seed=seed)
    ~~~~~
  File "/content/diffnaps/code/utils/gen_synth_datasets.py", line 148, in experiment1
    data, labels, gt_dict = gen_n_patterns_per_class(2,n_rows, n_cols,n_random,n_patterns,3,pattern_len=pattern_len)
    ~~~~~
  File "/content/diffnaps/code/utils/gen_synth_datasets.py", line 65, in gen_n_patterns_per_class
    gt_dict[i] = gen_patterns(range(0,n_cols),n_patterns, pattern_len)
    ~~~~~
  File "/content/diffnaps/code/utils/gen_synth_datasets.py", line 12, in gen_patterns
    return [sorted(random.sample(set(items), random.choice(range(*ran)))) for i in range(n)]
    ~~~~~
  File "/content/diffnaps/code/utils/gen_synth_datasets.py", line 12, in <listcomp>
    return [sorted(random.sample(set(items), random.choice(range(*ran)))) for i in range(n)]
    ~~~~~
  File "/usr/lib/python3.11/random.py", line 439, in sample
    raise TypeError("Population must be a sequence. "
TypeError: Population must be a sequence. For dicts or sets, use sorted(d).
```

# Exp1 – Escalabilidade – Numero de features

- Ele começa a rodar e não usa GPU
- Contudo, ao rodar localmente o processo tomou um kill, embora ele chegue a gerar os dados
- Foi estimado que seriam necessários 74 GB de RAM, o qual o grupo não tinha disponível

```
[2] !python code/exp1.py
```

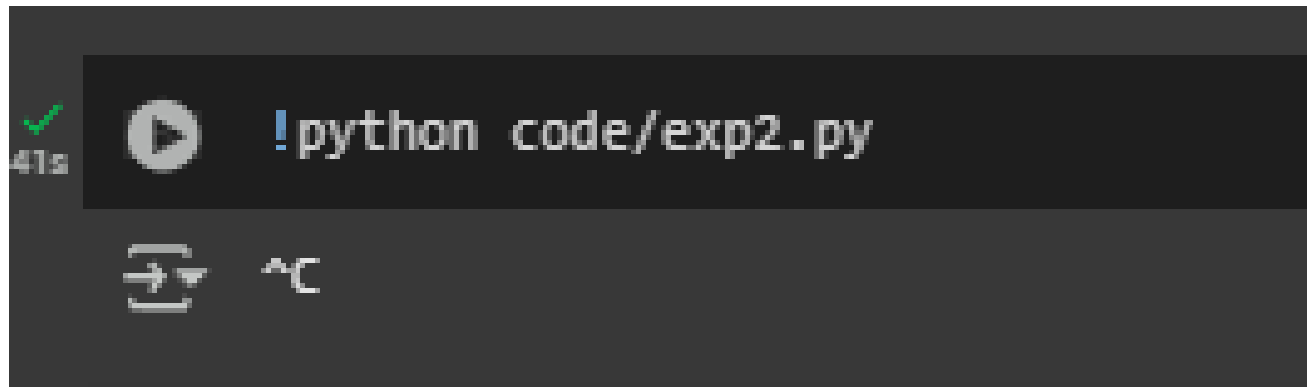
```
[100, 500, 1000, 5000, 10000, 15000, 20000, 25000, 50000, 1000000]
```

```
^C
```



# Exp2 – Escalabilidade - Número de Classes

- Ele também não usa GPU, contudo ele possui uma configuração do torch que também demanda cerca de 75 GB de RAM.

A screenshot of a terminal window with a dark background. On the left, there is a green checkmark icon and the text '41s'. In the center, there is a play button icon followed by the command '!python code/exp2.py'. At the bottom left, there are icons for file operations (a folder with an arrow) and a keyboard shortcut '^C'.

```
✓ 41s !python code/exp2.py
```

# Exp3 – Robustez e Ruído (aditivo)

- Finalmente funciona. Demora cerca de 16 minutos
- Gera uma nova pasta de resultados, com uma outra pasta chamada exp3 em que ele roda 5 testes diferentes gerando um csv com as métricas de avaliação mostradas no artigo

ncols	JD	SP	SR	F1	time
0.0	0.17556305974727027	0.4416961875843455	0.17556305974727027	0.2478557568871598	0.2915793498357137
10.0	0.17692461882046995	0.24750363860277652	0.17692461882046995	0.2054463911662293	0.29208837350209554
20.0	0.061992706992706995	0.23015873015873015	0.061992706992706995	0.09724921433188136	0.28734186887741087
30.0	0.08927039828532679	0.09574430199430199	0.08927039828532679	0.09239408500690943	0.2950247089068095
40.0	0.06764882813663302	0.10124135124135125	0.067648828136633	0.08110428676990858	0.28714914321899415
50.0	0.07635442988384164	0.14024413086913087	0.07635442988384165	0.09167994867794332	0.29337875843048095
60.0	0.1518909539340212	0.2884247448979592	0.15189095393402113	0.19898954208055525	0.28750317096710204
70.0	0.05818327505827505	0.10242424242424242	0.05818327505827505	0.07421044746867263	0.2946104129155477
80.0	0.10218043450802072	0.13434335272570566	0.10218043450802072	0.1160751086842017	0.28810928265253705
90.0	0.024511599511599515	0.06127899877899878	0.02451159951159951	0.03501657073085645	0.29300784269968666

16min



!python code/exp3.py



Test set: Average loss: 1430.459961, Recon Accuracy: 868294/1000000 (87%), Classification Accuracy: 177/200 (88%)

Train Epoch: 45 [0/1800 (0%)] Loss: 361.532776

Test set: Average loss: 1430.459961, Recon Accuracy: 868294/1000000 (87%), Classification Accuracy: 177/200 (88%)

Train Epoch: 46 [0/1800 (0%)] Loss: 364.267059

Test set: Average loss: 1443.131592, Recon Accuracy: 863823/1000000 (86%), Classification Accuracy: 178/200 (89%)

Train Epoch: 47 [0/1800 (0%)] Loss: 361.771759

Test set: Average loss: 1439.400513, Recon Accuracy: 863273/1000000 (86%), Classification Accuracy: 177/200 (88%)

Train Epoch: 48 [0/1800 (0%)] Loss: 362.348999

Test set: Average loss: 1437.562866, Recon Accuracy: 868449/1000000 (87%), Classification Accuracy: 179/200 (90%)

Train Epoch: 49 [0/1800 (0%)] Loss: 365.492401

Test set: Average loss: 1442.154541, Recon Accuracy: 866403/1000000 (87%), Classification Accuracy: 178/200 (89%)

Train Epoch: 50 [0/1800 (0%)] Loss: 364.092773

Test set: Average loss: 1441.456665, Recon Accuracy: 866594/1000000 (87%), Classification Accuracy: 179/200 (90%)

Encoder mean: tensor(0.0019)

Encoder: tensor(0.0191)

10 0.26663166701974295

##### 20 3 #####

tensor(6.4618e-06, device='cuda:0', grad\_fn=<MeanBackward0>)

Data Sparsity:

0.0432586

Train Epoch: 1 [0/1800 (0%)] Loss: 3016.811523

# Exp4 – Robustez e Ruído(destrutivo)

- Ele também chega a funcionar – roda em 14 minutos
- Ele gera os resultados de uma maneira similar ao exp3, mas com um csv muito maior com as colunas:  
ncols, JD\_mean, JD\_std, JD\_max, JD\_min, SP\_mean, SP\_std, SP\_max, SP\_min, SR\_mean, SR\_std, SR\_max, SR\_min, F1\_mean, F1\_std, F1\_max, F1\_min, time\_mean, time\_std, time\_max, time min.

ncols	JD_mean	JD_std	JD_max	JD_min	SP_mean	SP_std	SP_max	SP_min	SR_mean	SR_std	SR_max	SR_min	F1_mean	F1_std	F1_max	F1_min	time_mean	time_std	time_max	time min
0.0	0.1863401682197551	0.0455668524234418	0.2353455155955156	0.13417349991696859	0.4436767428433427	0.20634481875647082	0.6402777777777777	0.18648373983739835	0.1863401682197551	0.045566852423441793	0.2353455155955156	0.1341734999169686	0.2512639508007696	0.08192816752445832	0.33					
0.01	0.2106189731461477	0.05552609592018555	0.2815827502996497	0.12831396544631837	0.5149347700472748	0.2315590441552625	0.8369314436387607	0.29053775744952215	0.21061897314614772	0.055526095920185536	0.2815827502996497	0.1283139654463184	0.28008459914683853	0.07310989292769755	0.35					
0.05	0.1801301112117908	0.055199273771895645	0.23849357773531651	0.11267172577305817	0.3224758169415746	0.14348173297463301	0.5415178571428572	0.16232158541941152	0.1801301112117908	0.055199273771895645	0.23849357773531651	0.11267172577305816	0.22565631190528226	0.08036758095541702	0.32					
0.075	0.14521652019843684	0.020230669119163807	0.16130020920969196	0.11070724502380819	0.27626635968655744	0.05520914247743733	0.36322751322751323	0.22417582417582413	0.14521652019843684	0.020230669119163807	0.16130020920969196	0.11070724502380819	0.18174240322821258	0.02611106251438813	0.20					
0.1	0.12896967852766644	0.04327439506473073	0.17511928440142166	0.06677581497012795	0.2468189195445169	0.13877338816111917	0.3895444832944833	0.052885040655938485	0.12896967852766644	0.043274395064730714	0.17511928440142166	0.06677581497012795	0.16016496136439354	0.07042454786733099	0.23					
0.2	0.07730421625560609	0.05701031064880218	0.1572627189546526	0.016346153846153847	0.17625467028451391	0.12011551322490817	0.30782004830917875	0.0405982905982906	0.07730421625560609	0.05701031064880218	0.1572627189546526	0.016346153846153847	0.10320238480563529	0.07698413155338778	0.20					
0.3	0.014736640810170223	0.0065770926051992185	0.024583333333333332	0.007142857142857143	0.04776466248892719	0.023671246129175073	0.08194444444444444	0.019199608262108262	0.014736640810170223	0.0065770926051992185	0.024583333333333332	0.007142857142857143	0.022116383790880333	0.010363437056344044	0.03					
0.4	0.014312860086773132	0.010911332805442089	0.0302220695970696	0.005	0.04712468087468087	0.032625937627120405	0.09658882783882784	0.016666666666666666	0.014312860086773132	0.010911332805442089	0.0302220695970696	0.005	0.02105031034887025	0.01521293753708606	0.04					
0.5	0.002715267929469467	0.0038786936614907705	0.008350533195734432	0.0	0.00988235294117647	0.013935094230433917	0.029411764705882353	0.0	0.002715267929469467	0.0038786936614907705	0.008350533195734432	0.0	0.0042588670967877214	0.006065901431751655	0.01					
0.6	0.0009307359307359307	0.002081188810227077	0.004653679653679654	0.0	0.0047619047619047615	0.010647942749998997	0.023809523809523808	0.0	0.0009307359307359307	0.002081188810227077	0.004653679653679654	0.0	0.0015571247510411008	0.003481836792775338	0.00					

Show 10 per page

14min



!python code/exp4.py



```
Train Epoch: 1 [0/1800 (0%)] Loss: 330.000000
```

```
Test set: Average loss: 1427.233154, Recon Accuracy: 863370/1000000 (86%), Classification Accuracy: 185/200 (92%)
```

```
Train Epoch: 45 [0/1800 (0%)] Loss: 360.539642
```

```
Test set: Average loss: 1431.115967, Recon Accuracy: 864028/1000000 (86%), Classification Accuracy: 183/200 (92%)
```

```
Train Epoch: 46 [0/1800 (0%)] Loss: 358.926025
```

```
Test set: Average loss: 1428.757568, Recon Accuracy: 859898/1000000 (86%), Classification Accuracy: 182/200 (91%)
```

```
Train Epoch: 47 [0/1800 (0%)] Loss: 361.081543
```

```
Test set: Average loss: 1439.414917, Recon Accuracy: 853939/1000000 (85%), Classification Accuracy: 182/200 (91%)
```

```
Train Epoch: 48 [0/1800 (0%)] Loss: 364.946320
```

```
Test set: Average loss: 1434.665405, Recon Accuracy: 857037/1000000 (86%), Classification Accuracy: 183/200 (92%)
```

```
Train Epoch: 49 [0/1800 (0%)] Loss: 361.181641
```

```
Test set: Average loss: 1439.930054, Recon Accuracy: 865208/1000000 (87%), Classification Accuracy: 183/200 (92%)
```

```
Train Epoch: 50 [0/1800 (0%)] Loss: 363.180359
```

```
Test set: Average loss: 1435.262207, Recon Accuracy: 861512/1000000 (86%), Classification Accuracy: 183/200 (92%)
```

```
Encoder mean: tensor(0.0018)
```

```
Encoder: tensor(0.0189)
```

```
0.3 0.011904761904761902
```

```
##### 0.4 3 #####
```

```
tensor(6.4618e-06, device='cuda:0', grad_fn=<MeanBackward0>)
```

```
Data Sparsity:
```

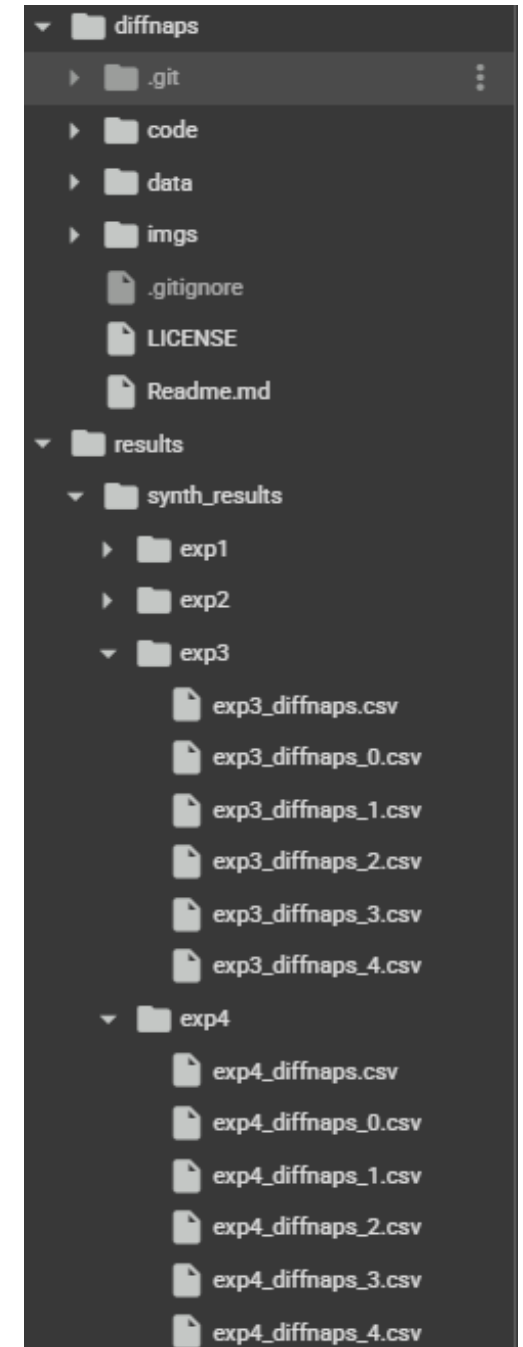
```
0.0390243
```

```
Train Epoch: 1 [0/1800 (0%)] Loss: 2991.600342
```

# Arquitetura final e Considerações

**É um código difícil e pesado**

- Não está 100% atualizado
- A base é difícil de recriar
- Pede recursos caros que torna reprodutibilidade difícil
- Mas ele roda, tá disponível na internet e os dados também.



# Referências

# Referências bibliográficas