Attention Models: da motivação às variantes modernas

Intuição geométrica, formulação matemática e aplicações

Rodrigo Petrus, Felipe Grael e Vivian Carvalho

19 de agosto de 2025

Table of Contents

1 Motivação e Histórico

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Redes Neurais Recorrentes

1 Motivação e Histórico

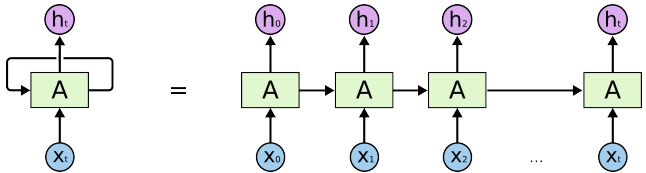


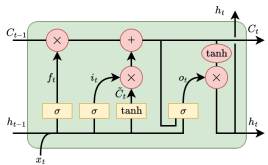
Figura: Redes Neurais Recorrentes [1]

- Bem adaptadas para dados sequenciais como séries temporais e texto
- Diferentes tipos de modelos: LSTM, GRU
- Diferentes arquiteturas: simples, bidirecional, encoder-decoder

[1] Figura de Christopher Olah

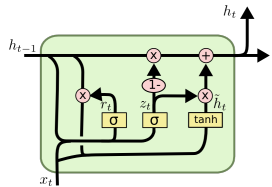
Tipos de camadas RNN

LSTM



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \qquad i_t = \sigma(W_i[h_{t-1}, x_t] + b_i),$$
$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \qquad h_t = o_t \odot \tanh(C_t)$$

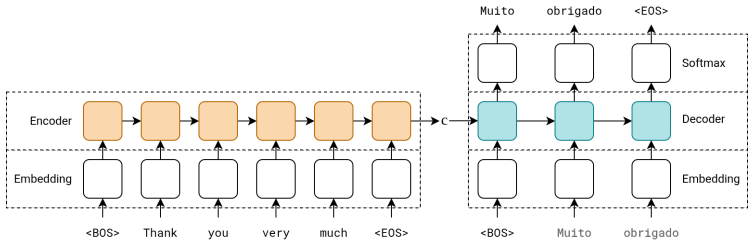
GRU



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Arquitetura Seq2Seq para Tradução

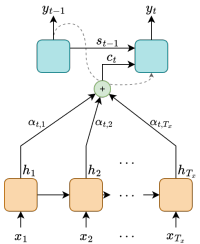
1 Motivação e Histórico

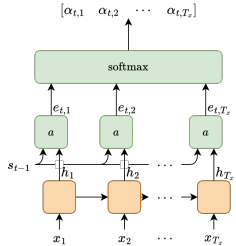


- **Gargalo**: toda a informação comprimida em $c = h_n$.
  - Processamento **sequencial** $\Rightarrow$ baixa paralelização.
  - **Dependências longas** ainda são difíceis (mesmo com portas).
  - **Gargalo do contexto** (vetor único) degrada qualidade em frases longas.

Atenção aditiva (Bahdanau [1])

1 Motivação e Histórico





- Cada etapa do decoder recebe um vetor de contexto c_i.
- O vetor de contexto é uma combinação ponderada dos estados do encoder.

- Escore de alinhamento avalia o quanto a palavra x_j é relevante para prever y_i.
- Calculado por uma rede neural feedforward.
- Resolve o gargalo de contexto e permit frases mais longas.

Atenção multiplicativa [2]

Três variantes de scoring:

$$e_{ij} = v^\top \tanh(W[s_j; h_i]) \quad \text{(concat, similar ao Bahdanau)}$$
$$e_{ij} = s_j^\top W h_i \quad \text{(general)}$$
$$e_{ij} = s_j^\top h_i \quad \text{(dot)}$$

- s_j: query → estado oculto do decoder.
- h_i: key/value → estado do encoder.
- **Concat:** aproxima-se da atenção aditiva de Bahdanau.
- **General:** bilinear, mais expressivo (aprende W).
- **Dot:** mais simples e rápido (nenhum parâmetro extra).

Nota: Podemos reinterpretar em termos modernos como $Q = s_j, K = h_i, V = h_i$.

Global vs Local Attention [2]

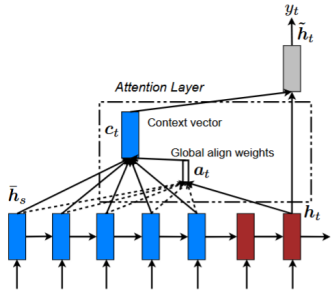


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

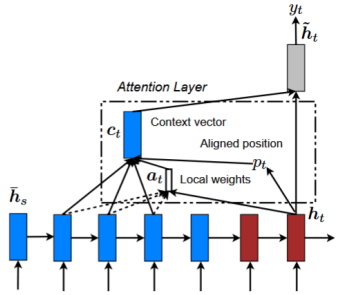


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a}_t$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

Self-Attention: dependências em paralelo

$$\mathrm{Att}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \quad Q = XW_Q,\ K = XW_K,\ V = XW_V.$$

- Calcula relações *entre todos os tokens* da mesma sequência, **em paralelo**.
- Multi-head:

$$\mathrm{MHA}(X) = \mathrm{Concat}(H_1,\ldots,H_h)\,W_O, \quad H_r = \mathrm{softmax}\left(\frac{Q_r K_r^\top}{\sqrt{d_k}}\right)V_r.$$

- Comparativo: RNN/LSTM exige n passos sequenciais; self-attention faz um passo paralelo com custo $\mathcal{O}(n^2)$.

Arquitetura pré-transformer

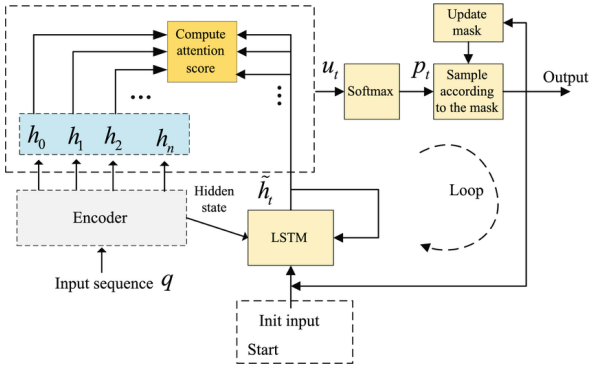


Figura: Exemplo de arquitetura pré-transformer (aditiva) [3]

Table of Contents

2 Transformers

- Motivação e Histórico
- Transformers
- Embeddings & Interpretações
- Treino & Hiperparâmetros
- Séries Temporais
- Language Models
- Vision Transformer
- Graph Attention
- Referências Bibliográficas

Attention Is All You Need [4]: nascendo o Transformer

2 Transformers

- **Remove** completamente a recorrência (sem LSTM).
- **Positional encodings** preservam ordem:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right).$$

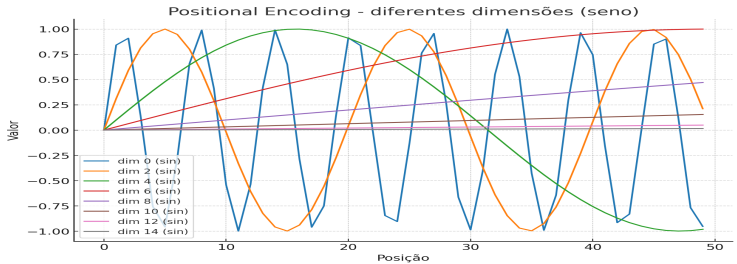


Figura: Exemplo de codificação posicional senoidal [4]

Bloco Transformer e arquitetura

- Cada **bloco Transformer** (pré-norm, forma comum):

$$Y = X + \mathrm{MHA}(\mathrm{LN}(X)),$$
$$Z = Y + \mathrm{FFN}(\mathrm{LN}(Y)), \quad \mathrm{FFN}(u) = W_2\,\phi(W_1 u + b_1) + b_2,$$

- Empilha-se vários blocos de atenção+FFN $\Rightarrow$ **arquitetura Transformer**.
- **Máscara causal** (para LMs) impede olhar o futuro:

$$\mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right), \quad M_{ij} = \begin{cases} 0, & j \le i \\ -\infty, & j > i \end{cases}$$

Arquiteturas Transformer e Aplicações

Encoder–Decoder
(Transformer original, 2017)

- Tradução automática
- Sumarização
- Diálogo
- Captioning

Encoder-only
(BERT, RoBERTa, DistilBERT)

- Classificação de texto
- NER (entidades)
- QA (extração de trechos)
- Análise semântica

Decoder-only
(GPT, LLaMA, etc.)

- Modelos de linguagem
- Geração de texto
- Completamento de prompts
- Story generation

Arquiteturas Transformer: encoder e decoder [4]

2 Transformers

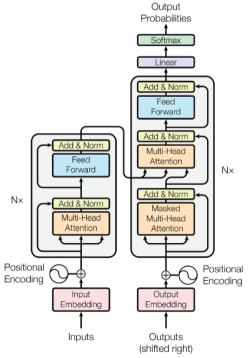


Figura: Transformer Encoder e Decoder

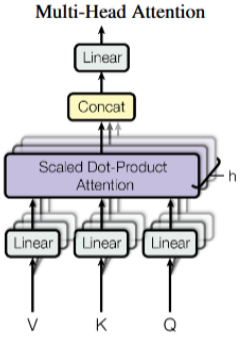


Figura: Multi-Head Attention

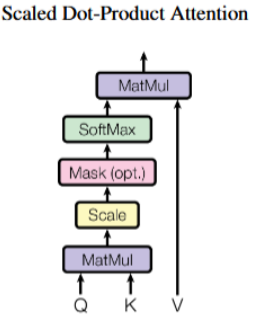


Figura: Multi-Head Attention

Resumo da evolução dos modelos

Linha do tempo (síntese):

- LSTM encoder–decoder: contexto único $c = h_n$ (gargalo).
- LSTM + **atenção** (Bahdanau/Luong): alívio do gargalo.
- **Self-attention**: dependências longas em paralelo.
- **Transformer**: atenção + posição + FFN; várias camadas empilhadas; sem LSTM.

Table of Contents

3 Embeddings & Interpretações

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Word/Subword Embeddings

3 Embeddings & Interpretações

- Estáticos (Word2Vec, GloVe) vs. Contextuais (ELMo, BERT)
- Subword (BPE/Unigram) para robustez morfológica
- Análogos em outras modalidades: patches (ViT), time2vec (TS), node2vec (grafos)

Atenção: projeções e compatibilidade

3 Embeddings & Interpretações

- **Projeções lineares**:
  $Q = XW_Q,\ K = XW_K,\ V = XW_V.$
- **Compatibilidade**: $e_i = \frac{\langle q, k_i \rangle}{\sqrt{d}}$ (ou cosseno se normalizar).
- **Pesos**:
  $\alpha_i = \mathrm{softmax}(e_i) \Rightarrow \alpha_i \geq 0,\ \sum_i \alpha_i = 1.$

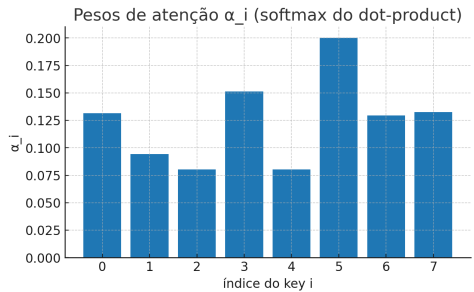
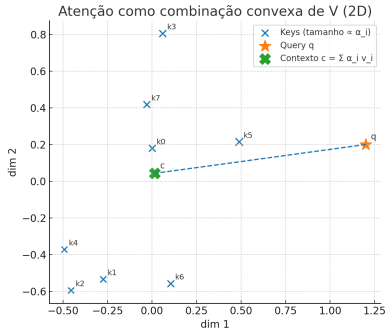


Figura: Pesos α_i (softmax do dot-product)

Interpretação Geométrica: contexto = barycenter

- **Contexto**: $c = \sum_i \alpha_i v_i \in \text{conv}\{v_i\} \Rightarrow$ combinação *convexa* dos values.
- **Geometria**: níveis de igual peso são hiperplanos ortogonais a q; pesos crescem exponencialmente com o alinhamento entre q e k_i.



Multi-head: múltiplas projeções/métricas

3 Embeddings & Interpretações

- Cada head aplica $(W_Q^{(h)}, W_K^{(h)}, W_V^{(h)})$ e induz uma métrica interna distinta: $\langle x, y \rangle_h = x^\top W_Q^{(h)} {W_K^{(h)}}^\top y$.
- Heads diferentes → diferentes pesos $\alpha^{(h)}$ e contextos $c^{(h)}$; a saída concatena/combina esses subespaços.

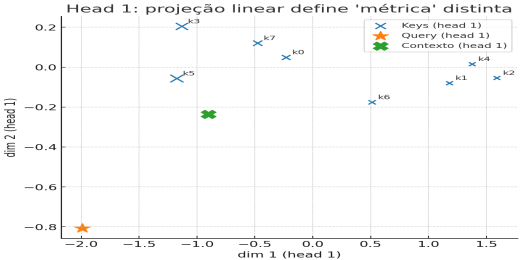


Figura: Head 1

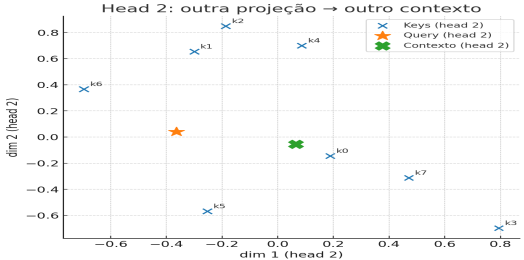


Figura: Head 2

Interpretação Matemática: Self-Attention

Fórmula central (scaled dot-product)

$$\mathrm{Att}(Q,K,V) \;=\; \mathrm{softmax}\!\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V$$

- Máscara causal/atencional conforme a tarefa
- Complexidade $\mathcal{O}(n^2)$ em tempo/memória
- Gradientes e saturação da softmax

Positional Encodings — Absolutos

- **Senoidais [4]:**

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right), \quad PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

  - Frequências diferentes em cada dimensão.
  - Permitem extrapolação por periodicidade.
  - Risco: aliasing (padrões que se repetem em posições distantes).

- **Aprendidos:**

$$z_{pos} = x_{pos} + P_{pos}, \quad P \in \mathbb{R}^{L_{\max} \times d}$$

  - Mais flexíveis e adaptados à tarefa.
  - **Limite**: não extrapolam para posições > $L_{\max}$.

Positional Encodings — Relativos e Bias

- **Shaw et al. (2018):**

$$e_{ij} = q_i^\top (k_j + r_{i-j})$$

  ou com bias b_{i-j}. Codifica **distâncias** em vez de posições absolutas.

- **T5 (Raffel et al., 2020):**

$$e_{ij} = q_i^\top k_j + B_{\text{bucket}(i-j)}$$

  - "Baldes" de distâncias (log-escalados).
  - Barato, simples, bom para extrapolação moderada.

- **ALiBi (Press et al., 2022):**

$$e_{ij} \; + \; m_h \, (i - j)$$

  - Inclinação linear por head.
  - Suporta janelas muito maiores sem retraining.
  - Praticamente custo zero.

Positional Encodings — RoPE e Impacto Prático

- **RoPE (Su et al., 2021):** aplica rotação dependente da posição:

$$\tilde{x}_{2i}, \tilde{x}_{2i+1} = \begin{cases} x_{2i}\cos\theta - x_{2i+1}\sin\theta \\ x_{2i}\sin\theta + x_{2i+1}\cos\theta \end{cases}, \quad \theta = \frac{pos}{10000^{2i/d}}$$

  - Produto $q^\top k$ passa a depender de $(pos_q - pos_k)$.
  - $\Rightarrow$ natural para codificar deslocamentos/ordem relativa.
- **Impacto prático:**
  - Absolutos aprendidos: limitados ao $L_{\max}$.
  - Senoidais: extrapolam mas sofrem com aliasing.
  - Relativos (T5, ALiBi, RoPE): melhor estabilidade para contextos longos.
- **Hoje**: ALiBi e RoPE são o padrão em LLMs de longo contexto.

Table of Contents

4 Treino & Hiperparâmetros

- Motivação e Histórico
- Transformers
- Embeddings & Interpretações
- Treino & Hiperparâmetros
- Séries Temporais
- Language Models
- Vision Transformer
- Graph Attention
- Referências Bibliográficas

Práticas de Treino

4 Treino & Hiperparâmetros

- AdamW, warmup + decaimento; label smoothing quando aplicável
- Regularização: dropout, stochastic depth, weight decay
- AMP/mixed precision, grad clipping, checkpointing
- Dados: curriculum, masking, augmentation (TS/ViT/GAT)

Hiperparâmetros Essenciais

- Profundidade, d_{model}, #heads, d_{ff}, dropout
- Comprimento de contexto, batch size, *LR schedule*
- Específicos: tokenização (LM), *patch size* (ViT), janela/patch (TS)

Qual tamanho ideal? (Scaling)

- **Leis de escala [5]:** desempenho cresce com ↑ dados, parâmetros e compute, até saturar.
- **Trade-offs:** muitos parâmetros + poucos dados → overfitting; muitos dados + poucos parâmetros → subutilização.
- **Tokens vs. parâmetros:** ideal quando #tokens ≈ múltiplos de #parâmetros [6].
- **Contexto:** janelas maiores ajudam, mas ganhos saturam.
- **Regra prática:** dimensione conforme dataset + compute + budget de inferência.

Resumo: não há tamanho ótimo universal — depende do equilíbrio entre dados, parâmetros e recursos disponíveis.

Table of Contents

5 Séries Temporais

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Atenção em Séries Temporais

5 Séries Temporais

- **Codificação temporal:** absolutos/relativos; *Time2Vec*; embeddings de calendário (hora/dia/sazonalidade).
- **Exógenas + cross-attention:** integrar variáveis externas (clima, preços, feriados) à série-alvo.
- **Contextos longos:** *patching* (janelas), sparsity e hashing; ganhos de janelas muito grandes tendem a saturar.
- **Por que atenção:** foca nos trechos relevantes do histórico e alinha padrões não estacionários/irregulares.
- **Tarefas:** previsão (forecasting), imputação (faltantes) e detecção de anomalias.

Modelos recentes (com código)

5 Séries Temporais

- Informer (AAAI'21) — atenção esparsa p/ sequências longas.
- Autoformer (NeurIPS'21) — auto-correlação + decomposição.
- PatchTST (ICLR'23) — *patching* / channel-independent.
- TimesNet (ICLR'23) — modelagem de variação temporal 2D.
- TSDiffusion (2023) - difusão para séries temporais.

Table of Contents

6 Language Models

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Language Models (LM)

- **Atenção causal** (*masked self-attention*): máscara triangular impede olhar o futuro.
- **Objetivo autoregressivo**: minimizar $-\sum_t \log p(x_t \mid x_{<t})$ (*next-token*).
- **Pré-treino** (corpus amplo, tarefa genérica) **vs. fine-tuning** (tarefa/estilo): *instruction tuning* (FLAN), RLHF (InstructGPT).
- **PEFT** (ajuste eficiente): *LoRA*, *Adapters* (congelar base + poucos params).
- **Métricas**: *perplexity* (surpresa média por token) e *downstream* (ex.: SuperGLUE, MMLU).

Language Models — Implementações (GitHub)

- karpathy/nanoGPT **(2022)** — GPT minimalista
- huggingface/transformers **(2019)** — biblioteca SOTA
- EleutherAI/gpt-neox **(2022)** — treino LLMs em larga escala
- meta-llama/llama **(2023)** — código/pesos Llama
- harvardnlp/annotated-transformer **(2018)** — Transformer anotado
- huggingface/trl **(2020)** — SFT, PPO/DPO, RLHF
- DeepSpeed-Chat **(2023)** — pipeline RLHF
- OpenRLHF/OpenRLHF **(2023)** — RLHF escalável
- huggingface/peft **(2023)** — PEFT (LoRA, QLoRA, etc.)
- adapter-hub/adapters **(2020)** — biblioteca de Adapters
- microsoft/LoRA **(2021)** — implementação LoRA oficial
- google-research/FLAN **(2021)** — dados p/ instruction tuning

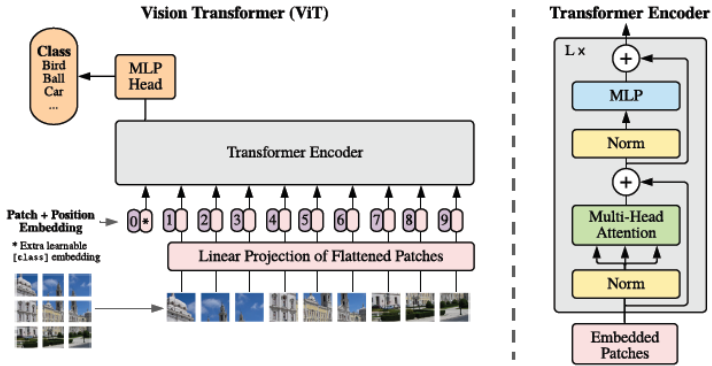
Table of Contents

7 Vision Transformer

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Vision Transformer (ViT) — Introdução

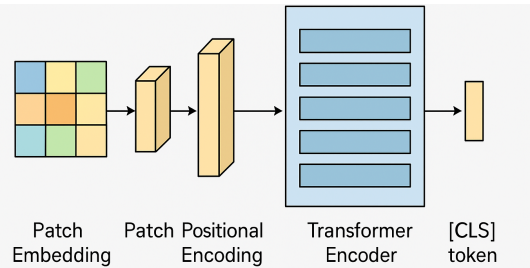
- CNNs dominaram visão computacional por anos (ResNet, EfficientNet, etc.).
- **ViT (Dosovitskiy et al., 2020):** adapta a atenção para imagens.
- Ideia central: dividir a imagem em **patches** $\rightarrow$ tokens $\rightarrow$ atenção global.



Arquitetura do Vision Transformer

7 Vision Transformer

- **Patch Embedding:** cada patch (ex: 16×16 pixels) é transformado em vetor.
- **Positional Encoding:** adiciona informação espacial aos tokens.
- **Transformer Encoder:** múltiplas camadas de *multi-head self-attention* + feedforward.
- **[CLS] token:** representa a imagem inteira para tarefas de classificação.



Resultados do Vision Transformer

7 Vision Transformer

- ViT pré-treinado em **JFT-300M** e **ImageNet-21k** supera CNNs de última geração.
- Comparação em benchmarks populares (ImageNet, CIFAR, Oxford Pets, Flowers, VTAB).
- Uso de **atenção global** traz ganhos em acurácia com menor custo de pré-treinamento que ResNets muito grandes.

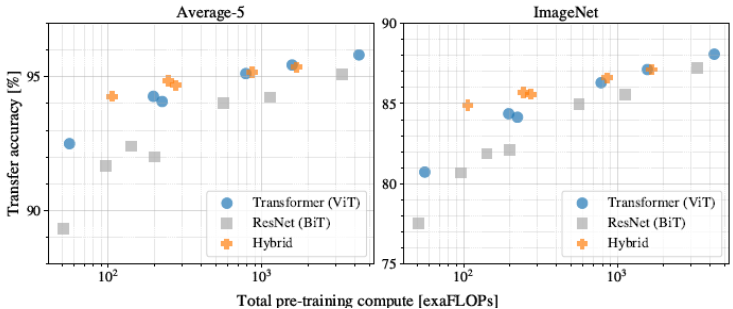
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Fonte: Dosovitskiy et al. (2020) *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*.

Pontos chave

7 Vision Transformer

- **Resultados:** ViT supera CNNs em grandes datasets (ex: ImageNet-21k, JFT).
- **Limitações:** precisa de muito dado e poder computacional para treinar do zero.
- **Avanços:** variantes mais leves como DeiT, Swin Transformer e modelos híbridos CNN+ViT.



Avanços do Vision Transformer

7 Vision Transformer

- **DeiT (Data-efficient Image Transformers)** Touvron et al., ICML 2021
  - Treino de ViTs menores com menos dados.
  - Uso de distilação de conhecimento.
- **Swin Transformer (Shifted Windows Transformer)** Liu et al., ICCV 2021
  - Atenção hierárquica em janelas deslizantes.
  - Mais eficiente para imagens grandes, detecção e segmentação.
- **Modelos Híbridos CNN+ViT** d'Ascoli et al., ICML 2021 (*ConViT*)
  - Combinação de convoluções locais + atenção global.
  - Inductive bias de CNN ajuda em datasets menores.

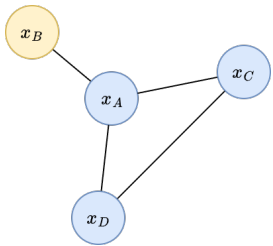
Table of Contents

8 Graph Attention

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Motivação: Graph Neural Networks

8 Graph Attention



- Cada nó possui um vetor de características x_i e um valor y_i a ser predito.

Rede Neural Simples

$$h_i = f(x_i W^T + b_i)$$

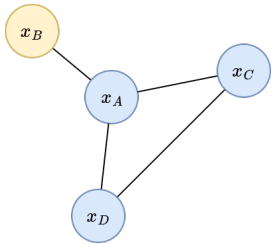
Graph Neural Network

$$h_i = f\left(x_i W^T + \sum_{j \in \mathcal{N}_i} x_j W^T\right)$$

- Agrega as features dos vizinhos de i ($\mathcal{N}_i$)
- Graph Convolutional Networks (GCNs) normalizam os pesos pelo grau dos nós.

Graph Attention Network [7]

8 Graph Attention



- Pondera cada vizinho com peso de atenção $\alpha_{i,j}$.
- Pode usar multi-head attention.

Graph Attention Networks (GATs)

$$\boldsymbol{h_i} = f\left(\boldsymbol{x_i W^T} + \sum_{j \in \mathcal{N}_i} \alpha_{i,j} \boldsymbol{x_j W^T}\right)$$

$$\alpha_{i,j} = \mathrm{softmax}_j(e_{i,j})$$

$$e_{i,j} = \mathrm{LeakyRelu}(a_{i,j})$$

$$a_{i,j} = \boldsymbol{W_{att}^T}[\boldsymbol{W x_i}, \boldsymbol{W x_j}]$$

Graph Attention Network [7]

8 Graph Attention

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

	Transductive		
Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	**79.0** $\pm 0.3\%$
GAT (ours)	**83.0** $\pm 0.7\%$	**72.5** $\pm 0.7\%$	**79.0** $\pm 0.3\%$

Evoluções de GATs

- **GATv2** [8]: atenção dinâmica (mais expressiva).
- **TGAT** [9]: Aprendizado de representação para grafos temporais

Table of Contents

9 Referências Bibliográficas

- ▶ Motivação e Histórico
- ▶ Transformers
- ▶ Embeddings & Interpretações
- ▶ Treino & Hiperparâmetros
- ▶ Séries Temporais
- ▶ Language Models
- ▶ Vision Transformer
- ▶ Graph Attention
- ▶ Referências Bibliográficas

Referências Bibliográficas

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, 2015.

[3] S. Gu and Y. Zhuang, "Method for solving constrained 0-1 quadratic programming problems based on pointer network and reinforcement learning," *Neural Computing and Applications*, vol. 35, 2022.

Referências Bibliográficas

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.

[6] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, J. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, “Training compute-optimal large language models,” *arXiv preprint arXiv:2203.15556*, 2022.

Referências Bibliográficas

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[8] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[9] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *ArXiv*, vol. abs/2002.07962, 2020.