# EECS 553 Lecture 15
# Machine Learning (ECE)
# [class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, October 22, 2026, 12:35

F26

# Logistics

© 2026 Fessler

- HW5 due Fri. Oct. 23
- HW6 due Wed. Nov. 4
- Mid-semester course feedback due today!
- Great Lakes Slurm account created (per student limits)
- Presentation schedule Piazza poll (now!)

Some slides from Prof. Samet Oymak may use different notation.

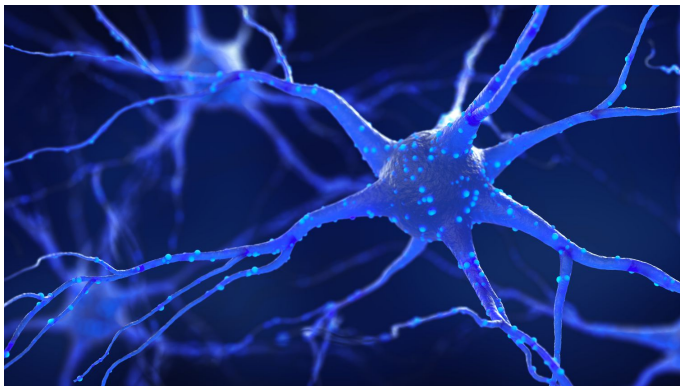
# Recap / Outline

© 2026 Fessler

- Recap
  - Backpropagation for NN training
  - CNN models
  - Read rest of CNN notes from I14 (no math)
- Today
  - Deep learning recap
  - Feature learning
  - Self-supervised learning
    - Pretext tasks
    - Self-supervised learning with images
    - Self-supervised NLP training
    - Word prediction (GPT)
  - Sequence modeling
  - Transformer / attention

# Motivation: abstract neuron model

© 2026 Fessler



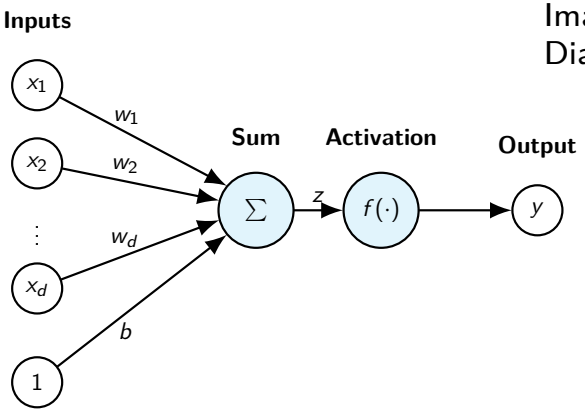



Image courtesy Getty images.
Diagram courtesy Gemini.

Inspired by **biological neurons**, an **artificial neuron** "activates" when the correlation (inner product) between the input and a pattern exceeds some threshold:

$$y = \text{threshold}(\boldsymbol{w}^\top \boldsymbol{x} + b) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b),$$

where $\sigma(\cdot)$ is called the **activation function**.
Recall **Cauchy-Schwarz inequality** from linear algebra:

$$\boldsymbol{w}^\top \boldsymbol{x} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle \leq \|\boldsymbol{w}\|_2 \, \|\boldsymbol{x}\|_2 \,,$$

with equality when

# Feedforward networks: Recap

© 2026 Fessler

- A **neural network** (**NN**) (or artificial NN) is a **prediction function** loosely inspired by **biological neural networks**.
- The network generally has linear computation between **layers** of the network and nonlinearities at nodes.
- Feedforward networks do not have any **feedback** connections.

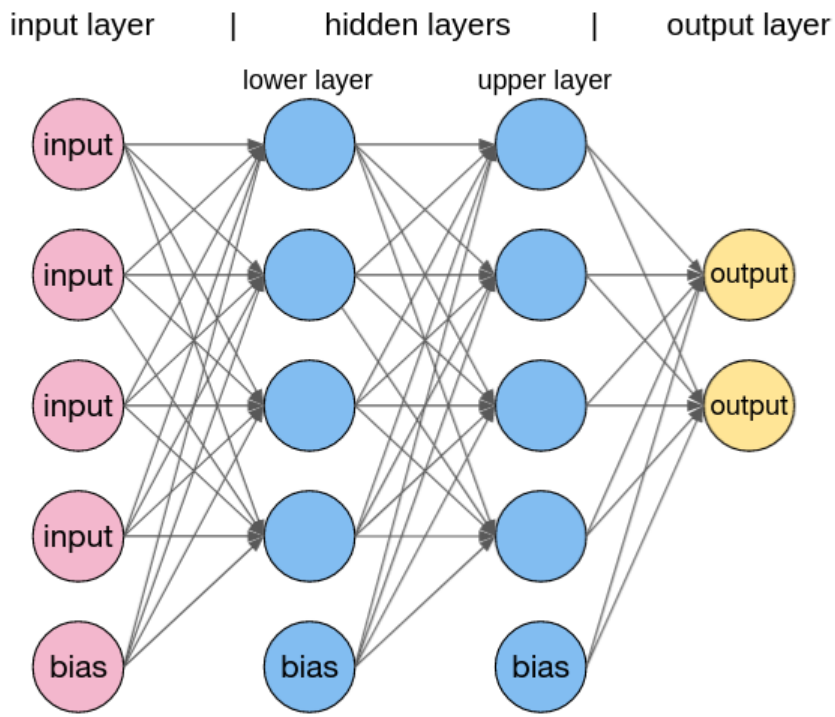



Figure courtesy [2].

# CQ: Feedforward networks

© 2026 Fessler

Q15.1 What does a user specify (before training) in a general **feed-forward neural network** with $L$ layers?

A: Number of hidden layers

B: Dimensions of all weight matrices

C: Connections between every pair of layers (all $\binom{L}{2}$ pairs)

D: Mathematical form of the activation function(s)

E: All of the above

# Feedforward network: Recap of Model

© 2026 Fessler

- For data $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d \times \mathbb{R}^k$, one such network model is:
  $\boldsymbol{y} = \sigma_3\big(\boldsymbol{W}^{(3)}\sigma_2\big(\boldsymbol{W}^{(2)}\sigma_1\big(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}_1\big) + \boldsymbol{b}_2\big) + \boldsymbol{b}_3\big),$
  where $\{\sigma_l\}$ are nonlinear functions (applied element-wise!) and $\{\boldsymbol{b}_l\}$ are (learned) bias vectors.
- We discussed using **backpropagation** to train all the parameters $\boldsymbol{\theta}$ of such networks.

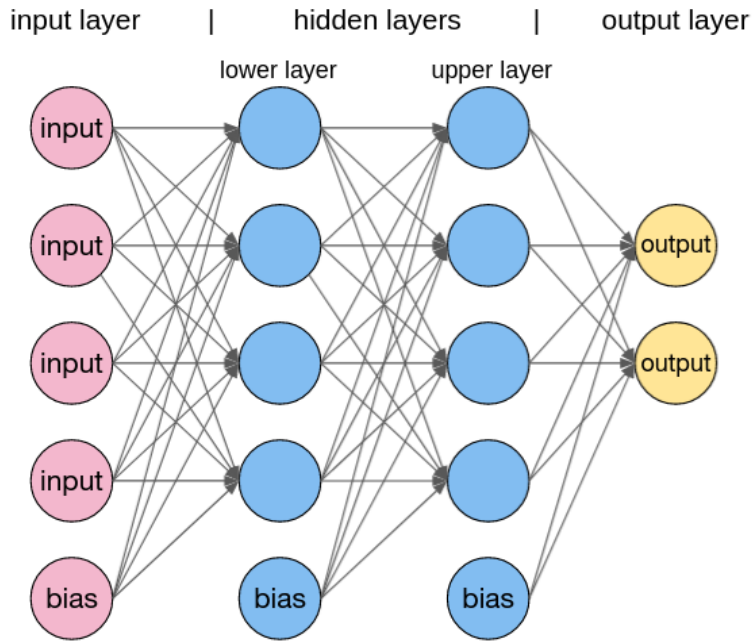
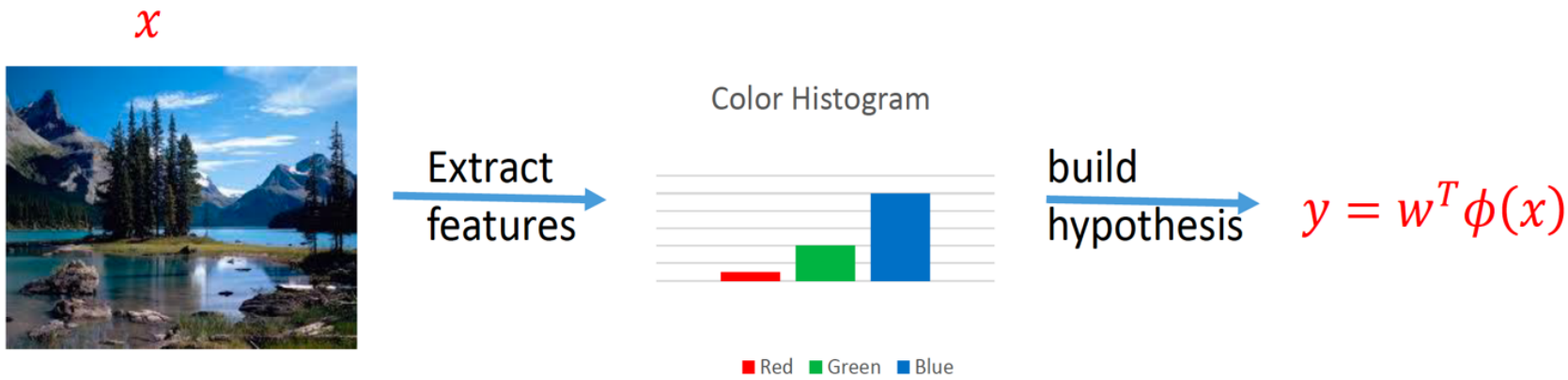



Figure courtesy [2].

- Traditional signal processing (aka pattern recognition) used "hand-crafted" features (like wavelet coefficients) as input to a classifier, whereas NN models *learn* features from data.

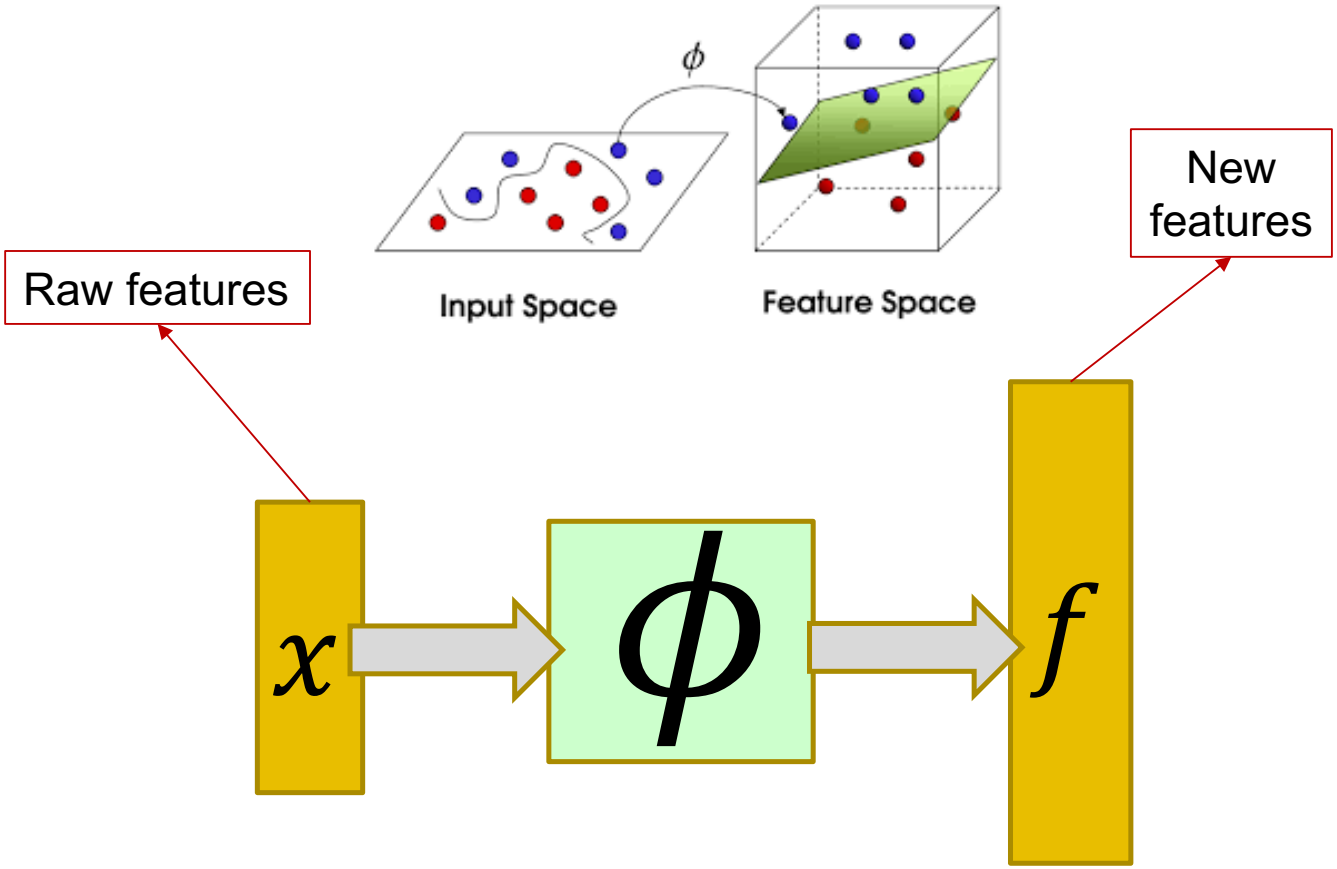



This figure and the next several slides are courtesy Samet Oymak, some of which come from Stanford's CS231n class slides (DL for CV).

- Traditional methods use hand-crafted features that might be quite nonlinear functions $\Phi(\boldsymbol{x})$ of the "raw" signal $\boldsymbol{x}$, followed by a "linear" classifier.
- For example, classical speech recognition systems used tools like the **spectrograms** formed by the **short-time Fourier transform** (**STFT**), followed by further processing using statistical models like **hidden Markov models** [3].

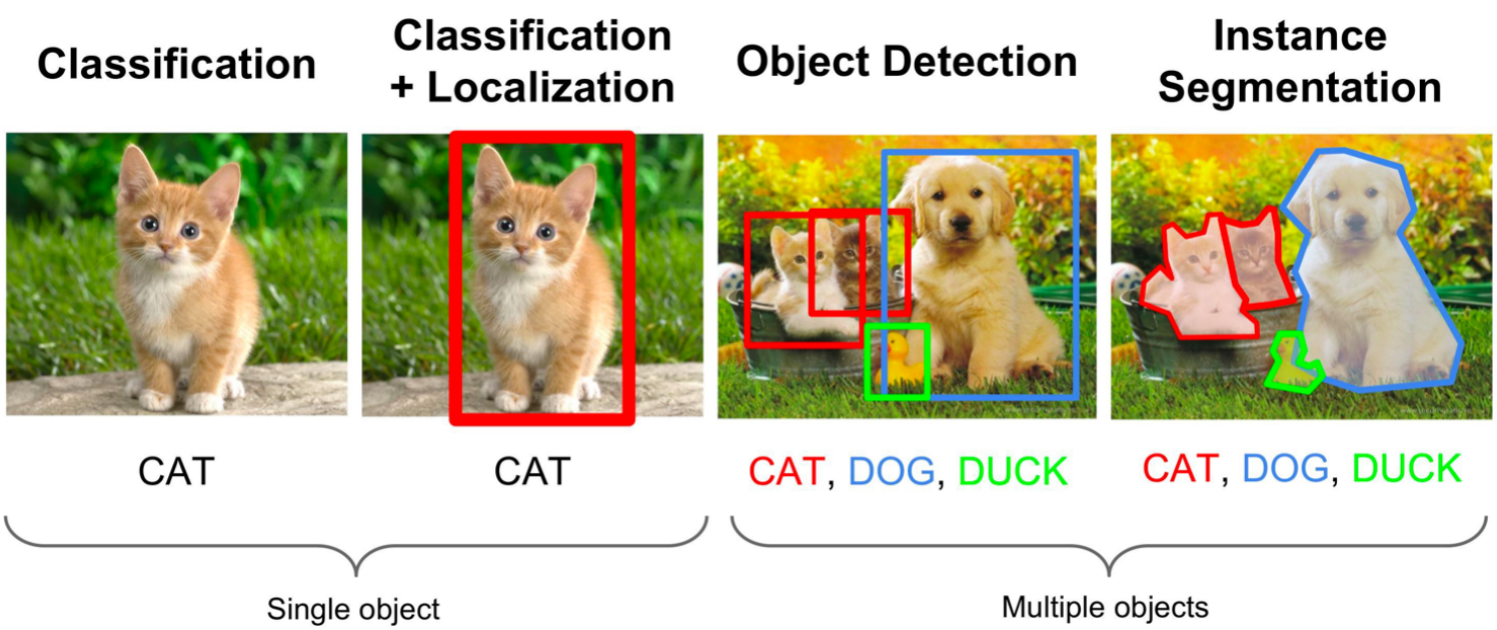
# Feature learning: Lifting

© 2026 Fessler




- Kernels correspond to specific nonlinear lifting functions $\mathbf{\Phi}(\cdot)$.
- Training a neural network prediction function can be interpreted as learning *from data*

A given type of data (*e.g.*, digital images) may be used in many applications:

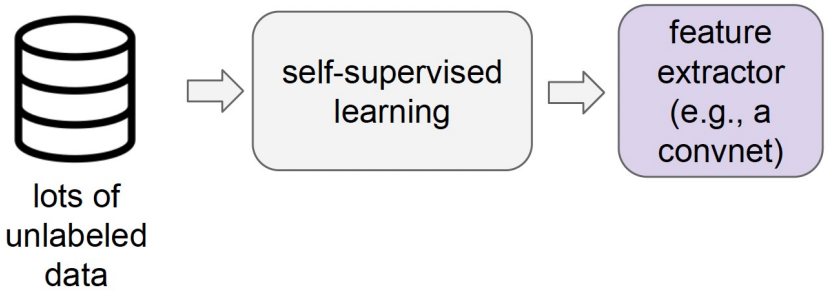



- All of these image analysis tasks would benefit from good image feature extractors.
- Traditionally one trains separate NN models for each such task.
- Can we build such feature extractors without labels?
- Motivations for

Start of (modified) slides from Prof. Samet Oymak

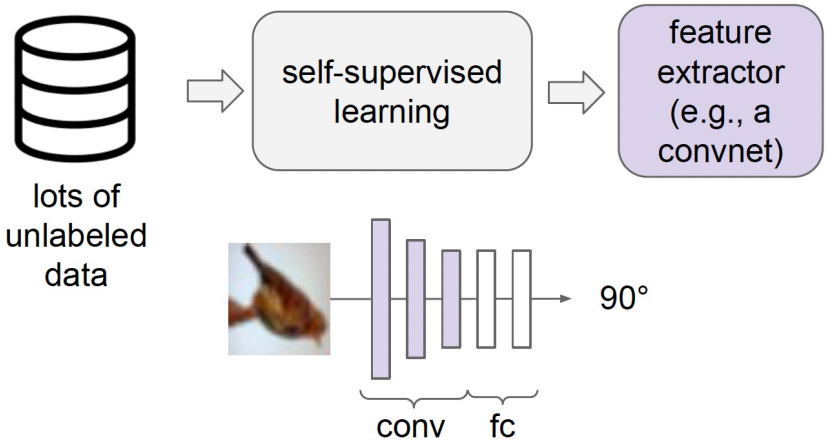
# Self-supervised learning pipeline: Goal

© 2026 Fessler




- Here the goal of **self-supervised learning** is to learn an useful **feature extractor**, *i.e.*, a function $\boldsymbol{\Phi}(\boldsymbol{x})$ typically represented by a NN model, for later use in many applications.
- From (widely available) unlabeled data.

# Self-supervised learning pipeline: Pretext tasks © 2026 Fessler

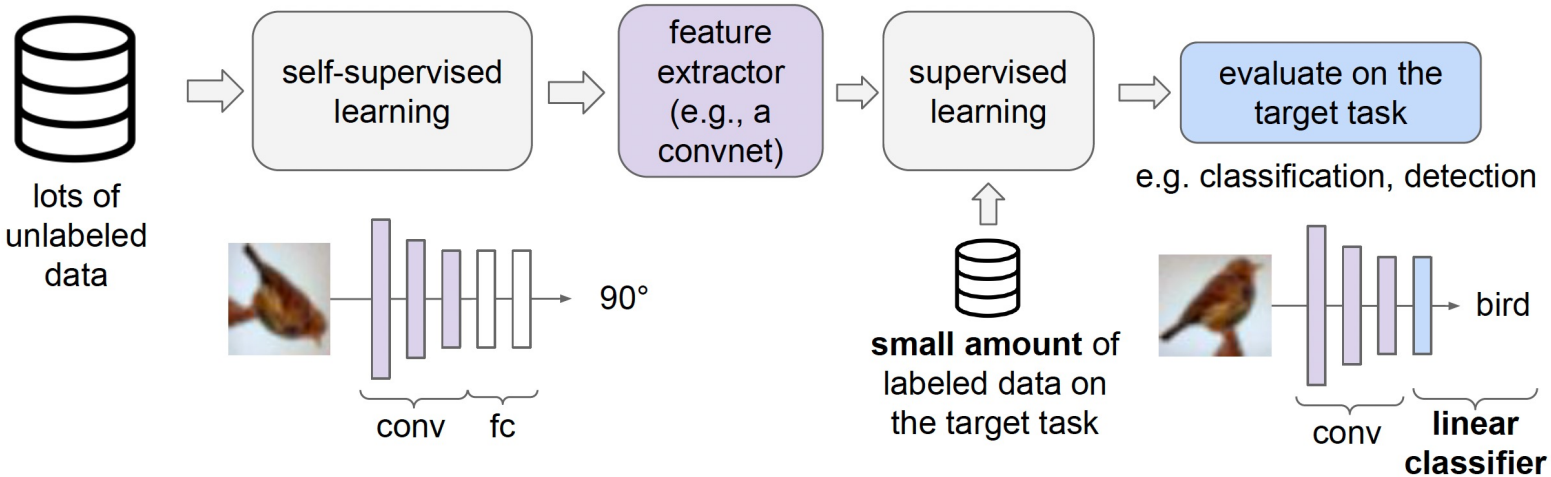



1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

- Focus on **autoassociative self-supervised learning** where model is trained to reproduce (some version of) the input.
  - images:
  - written language: sentences with
- These pretext tasks are

# Self-supervised learning pipeline: Deployment

© 2026 Fessler




- The resulting **feature extractor** could be called a **foundation model**: "any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (*e.g.*, fine-tuned) to a wide range of downstream tasks." Bommasani *et al.*, 2021
- Can the foundation model, or fine-tune it too (later).

# Self-supervised pretext tasks: Images

© 2026 Fessler

Example: learn to predict image transformations / complete corrupted images

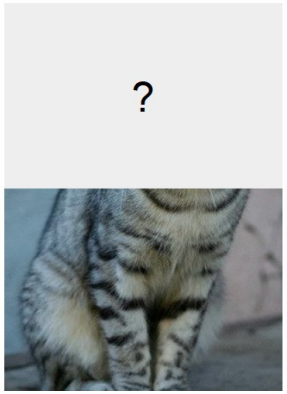
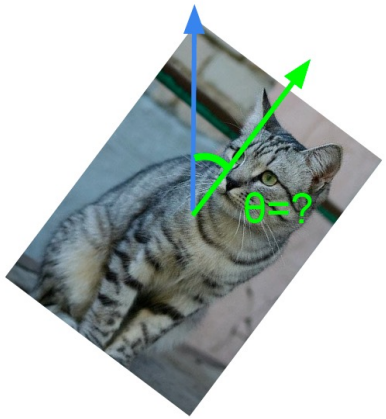
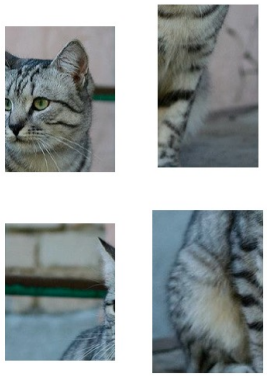



image completion




rotation prediction



"jigsaw puzzle"



colorization

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

Other options: adding noise, down-sampling (super-resolution), ...

# How to evaluate a self-supervised learning method? © 2026

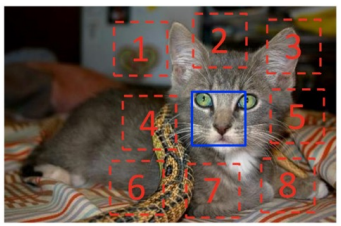
Fessler

- We usually do not care much about the performance of the self-supervised learning task, *e.g.*, whether the model learns to predict image rotation perfectly is relatively unimportanty.
- Evaluate the learned feature encoders on downstream *target* tasks.
- In a sense, pretext tasks are somewhat akin to surrogate loss functions.

# Applicable in diverse settings

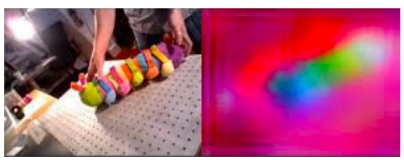
© 2026 Fessler

computer vision




Doersch et al., 2015

robot / reinforcement learning



Dense Object Net (Florence and Manuelli et al., 2018)

language modeling



**Language Models are Few-Shot Learners**

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

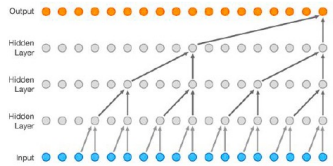
OpenAI

**Abstract**

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

GPT3 (Brown, Mann, Ryder, Subbiah et al., 2020)

speech synthesis




Wavenet (van den Oord et al., 2016)

...

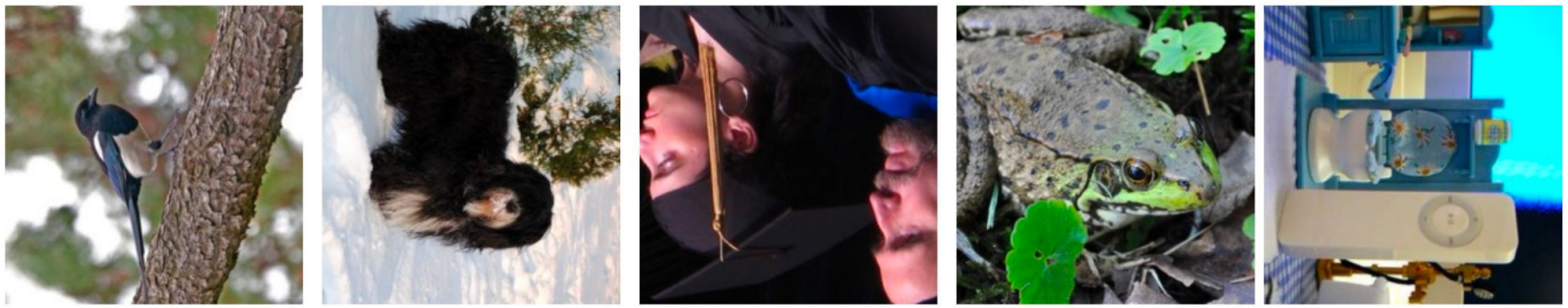
Of course there are many open research problems, *e.g.*, reliably solving math problems using LLMs.

- **Autoassociative learning** with **pretext tasks**
  - Vision: image transformations like rotation, inpainting, rearrangement, coloring.
  - NLP: predict missing or subsequent words in a sentence.
  - Audio:
    - mask part of signal or spectrogram and predict missing part
    - predict subsequent part of audio signal
    - estimate time gap between two audio clips from one stream
    - chop and shuffle signal, and classify whether order is correct or solve the "jigsaw puzzle" of rearranging
    - add noise or change pitch or shift speed: classify whether same or not
    - audio-visual correspondence (AVC) between video track an audio track (are they sychronized or mismatched?)

- **Contrastive representation learning**
  - Intuition: A transformation of an image should be more similar to the original image compared to another random image
  - SimCLR method [4]

# Image pretext task: predict rotations

© 2026 Fessler

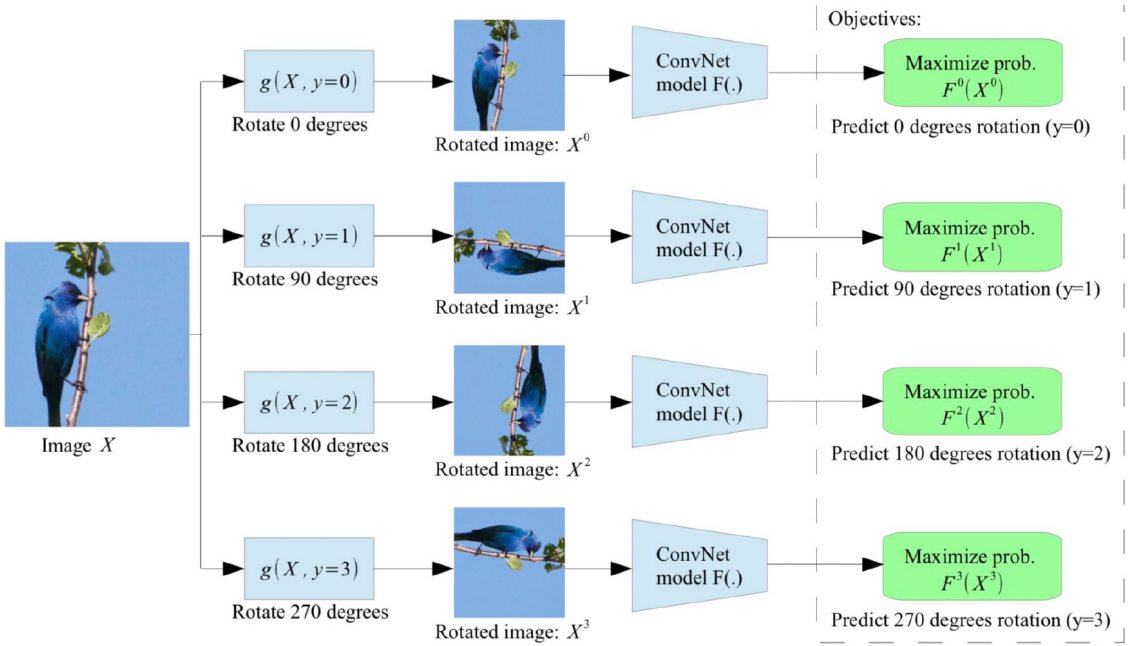


90° rotation | 270° rotation | 180° rotation | 0° rotation | 270° rotation

**Hypothesis:** a model could recognize the correct rotation of an object only if it has the “visual commonsense” of what the object should look like unperturbed.

# Image pretext task: predict rotations

© 2026 Fessler




Self-supervised learning by rotating the entire input images.

The model learns to predict which rotation is applied (4-way classification)

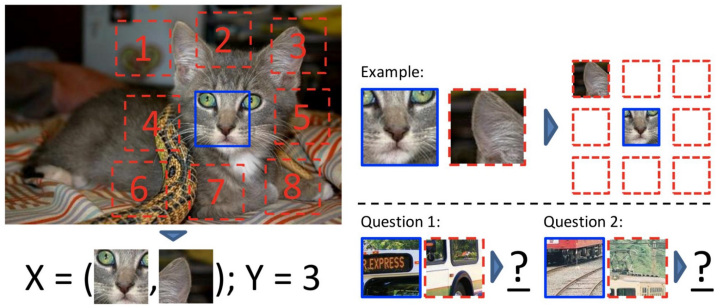
(Image source: Gidaris et al. 2018)

- 4-way rotation classification is clearly more feasible than arbitrary rotation (regression)
- How would you do this with classical signal processing methods?

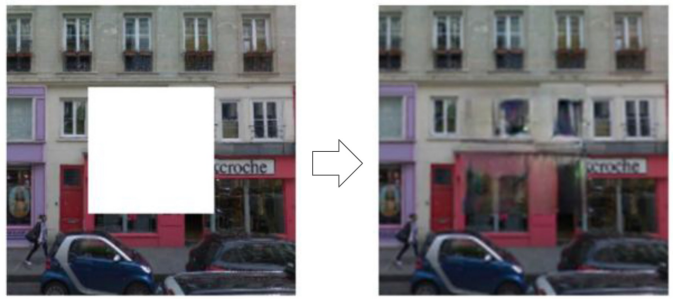
# Other vision pretext tasks

© 2026 Fessler

Pretext task: predict relative patch locations



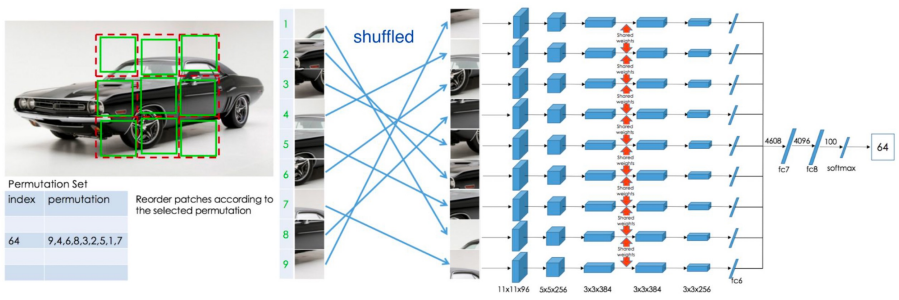

Pretext task: predict missing pixels (inpainting)



*by Inpainting* (Pathak et al., 2016)

Source: Pathak et al., 2016

Pretext task: solving "jigsaw puzzles"




(Image source: Noroozi & Favaro, 2016)

Two kinds of image **inpainting**:

- Easier: missing random pixels (*cf* matrix completion)
- Harder: missing large contiguous regions (*e.g.*, rectangles)

- Contrastive (self-supervised) representation learning intuition: A transformation of an image should be more similar to the original image compared to another random image

"A simple framework for **contrastive learning** of visual representations" (SimCLR paper by Chen, et al., ICML 2020)

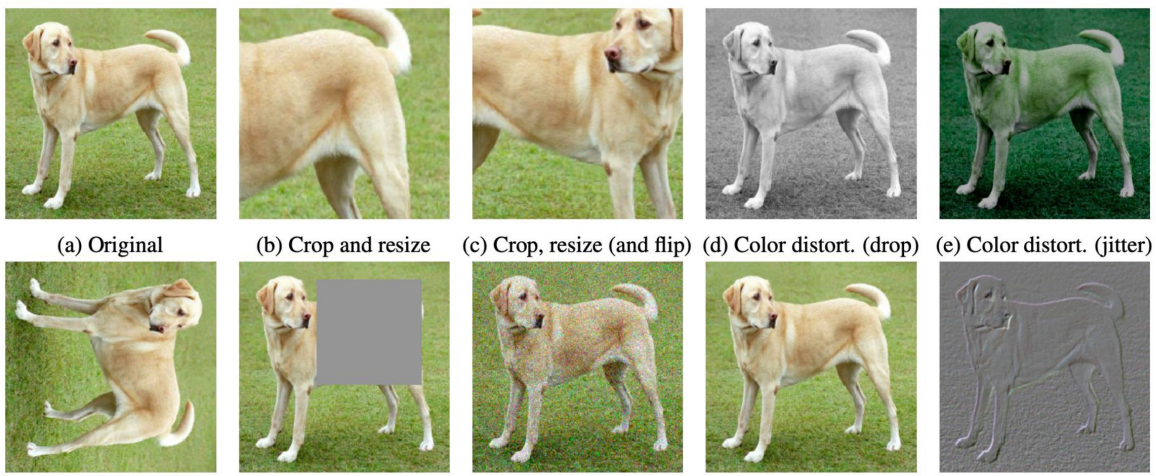



Figure from [5].

"A linear classifier trained on self-supervised representations learned by SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over previous ..., matching the performance of a supervised ResNet-50. When fine-tuned on only 1% of the labels, we achieve 85.8% top-5 accuracy, outperforming AlexNet with 100× fewer labels."

# Other learning frameworks

© 2026 Fessler

- The tabular data foundation models TabPFN TabFM are not trained with self-supervised learning.
- Instead, they are trained using a technique called Prior-Data Fitted Networks (PFNs) via supervised meta-learning (in-context learning) on millions of synthetically generated datasets.
- The core training mechanism is Supervised Meta-Learning / In-Context Learning (ICL): The model is fed an entire synthetic dataset (inputs + true labels) as a prompt context to approximate the Bayesian posterior predictive distribution.
- The "Task" is Direct End-to-End Prediction: The model learns the meta-task of "how to fit a machine learning model" in a single forward pass.

# Language modeling: Overview

© 2026 Fessler

- Example downstream NLP tasks:
  - Classify sentiment of a sentence
  - Translate a sentence from English to Spanish
  - Solve a math problem
  - Generate code
  - Write an document on a topic
  - ...
- Pretext tasks aim to learn the structure of language:
  - Predict next word (or words) in a sentence
  - Predict (random) missing (masked) words in a sentence

# Pretext tasks for NLP

© 2026 Fessler

- Next word prediction examples:
  - $\boldsymbol{X}$ = "Machine learning is a subfield of artificial", $Y$ = "intelligence"
  - $\boldsymbol{X}$ = "2 + 3 =", $Y$ = "5"
- Predicting missing words (a.k.a. masked language modeling)
  - $\boldsymbol{X}$ = "Machine learning is a subfield of [M] intelligence", $Y$ = "artificial"
  - $\boldsymbol{X}$ = "2 + [M] = 5", $Y$ = "3"
- We use the mathematical notation and terminology of "feature" $\boldsymbol{X}$ and "label" $Y$ above, but this is "simply" **autoassociative learning** using two "hand-crafted" pretext tasks.
  No labeled data is needed!
  Just need tons of text

# Word prediction (GPT architectures)

© 2026 Fessler

- Generates sequences where each token is predicted based on the previously generated tokens
- Uses causal attention masking to ensure the causality i.e. future info can't help predict the next token
- GPT stands for "generative pretrained transformer"

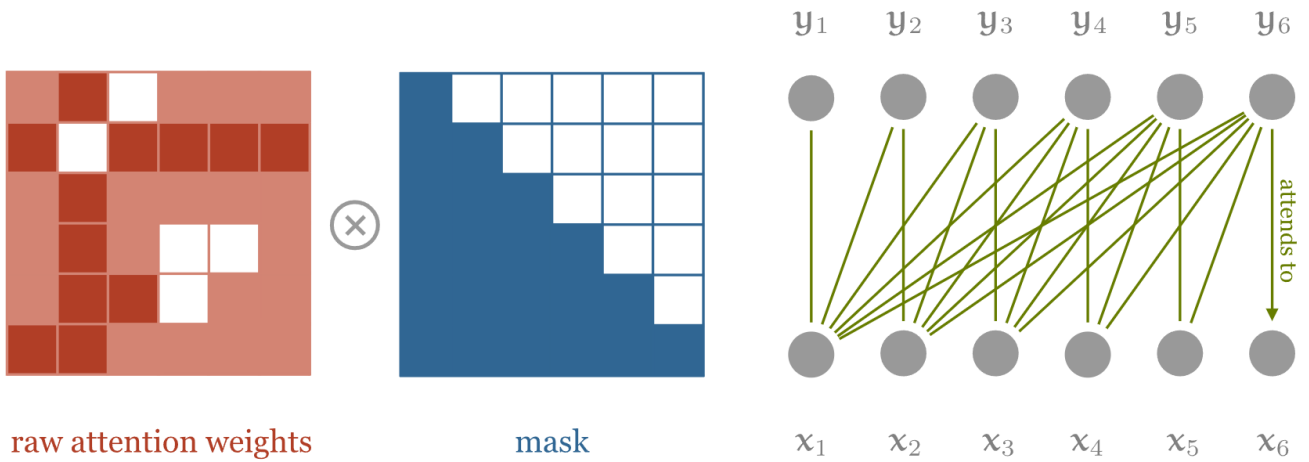



Figure source: https://peterbloem.nl/blog/transforme

Here the "⊗" denotes

# Design choices of GPT models

© 2026 Fessler

| Transformer Model | Parameters Count | Num Layers | Embedding Dimensions | Context/Sequence Length |
|---|---|---|---|---|
| **GPT (Original)** | ~117M | 12 | 768 | 512 |
| **GPT-2 Small** | ~117M | 12 | 768 | 1024 |
| **GPT-2 Medium** | ~345M | 24 | 1024 | 1024 |
| **GPT-2 Large** | ~774M | 36 | 1280 | 1024 |
| **GPT-2 XL** | 1.5B | 48 | 1600 | 1024 |
| **GPT-3** | 175B | Up to 96 | Up to 12288 | 2048 |
| **GPT-3.5** | 175B | - | - | 4096 |

- GPT 4 and GPT 5 are not necessarily larger in parameter counts, but NN architecture evolves. **Meta's Llama 4** has parameters (for text and images) [6].
- Embedding dimension is the dimension (our $d$) of the $\boldsymbol{x}$ representing each **token** (roughly a word in a vocabulary)
- Context length grew slowly initially, for complexity reasons that we will see shortly.

Last slide from Prof. Oymak

# Next word prediction: Probability foundation

© 2026 Fessler

- We wish to find the model that maximizes data likelihood:

- Use the **chain rule** for conditional probability to write

$$\mathrm{P}\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N; \boldsymbol{\theta}\} =$$

- Maximizing this is equivalent to

$$\arg\min_{\boldsymbol{\theta}}$$

  - Finding $\boldsymbol{\theta}$ that leads to the most accurate predictions of individual next words optimizes the overall sequence likelihood.

- Pixels in an image are a static 2D array, whereas language (both text and audio) involves a *sequence*.
- Sequence modeling has a long history in statistics, signal processing, control, ... and now in deep learning.
- It remains and active research area, with linear algebra as a key.

**Table 1. Unification of linear sequence mixers L in matrix form and sequential form.**

| Family | Mixer Model | Matrix Form | Sequential Form |
|---|---|---|---|
| Transformer | Softmax attention [11] | LV<br>$\mathbf{L} = \mathrm{norm}(\exp(\mathbf{QK}^\top) \odot \mathbf{M})$ | $\mathbf{y}_i^\top = \sum_{j=1}^{i} \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_j)}{\sum_{p=1}^{i} \exp(\mathbf{q}_i^\top \mathbf{k}_p)} \mathbf{v}_j^\top$ |
| Linear Transformer | Linear attention [16] | LV<br>$\mathbf{L} = \boldsymbol{\phi}(\mathbf{Q})\boldsymbol{\phi}(\mathbf{K})^\top \odot \mathbf{M}$ | $\mathbf{H}_i = \mathbf{H}_{i-1} + \boldsymbol{\phi}(\mathbf{k}_i)\mathbf{v}_i^\top$<br>$\mathbf{y}_i^\top = \boldsymbol{\phi}(\mathbf{q}_i)^\top \mathbf{H}_i$ |
| | + Normalized | LV<br>$\mathbf{L} = \mathrm{norm}(\boldsymbol{\phi}(\mathbf{Q})\boldsymbol{\phi}(\mathbf{K})^\top \odot \mathbf{M})$ | $\mathbf{z}_i = \mathbf{z}_{i-1} + \boldsymbol{\phi}(\mathbf{k}_i)$<br>$\mathbf{y}_i^\top = \frac{\boldsymbol{\phi}(\mathbf{q}_i)^\top \mathbf{H}_i}{\boldsymbol{\phi}(\mathbf{q}_i)^\top \mathbf{z}_i}$ |
| | RFA [18], RetNet [19] | LV<br>$\mathbf{L}_{ij} = \mathbf{q}_i^\top \mathbf{k}_j \cdot (\mathbb{1}_{j \le i} \cdot \gamma^{i-j})$ | $\mathbf{H}_i = \gamma \mathbf{H}_{i-1} + \mathbf{k}_i \mathbf{v}_i^\top$<br>$\mathbf{y}_i^\top = \mathbf{q}_i^\top \mathbf{H}_i$ |
| | GLA [20] | LV<br>$\mathbf{L}_{ij} = \left(\mathbf{q}_i \prod_{k=1}^{i} \lambda_k\right)^\top \left(\prod_{k=1}^{j} \lambda_k^{-1} \mathbf{k}_j\right)$ | $\mathbf{H}_i = \lambda_i \odot \mathbf{H}_{i-1} + \mathbf{k}_i \mathbf{v}_i^\top$<br>$\mathbf{y}_i^\top = \mathbf{q}_i^\top \mathbf{H}_i$ |
| SSM | Mamba-2 [32] | LU<br>$\mathbf{L}_{ij} = \mathbf{c}_i^\top \left(\prod_{k=j+1}^{i} \lambda_k\right) \mathbf{b}_j$ | $\mathbf{H}_i = \lambda_i \mathbf{H}_{i-1} + \mathbf{b}_i \mathbf{u}_i^\top$<br>$\mathbf{y}_i^\top = \mathbf{c}_i^\top \mathbf{H}_i + \mathbf{D}\mathbf{u}_i^\top$ |
| | Mamba [31] | LU *(tensor contraction)*<br>$\mathbf{L}_{ij} = \mathbf{C}_i^\top \left(\prod_{k=j+1}^{i} \boldsymbol{\Lambda}_k\right) \mathbf{B}_j$ | $\mathbf{H}_i = \boldsymbol{\lambda}_i \odot \mathbf{H}_{i-1} + \mathbf{b}_i \mathbf{u}_i^\top$<br>$\mathbf{y}_i^\top = \mathbf{c}_i^\top \mathbf{H}_i + \mathbf{D}\mathbf{u}_i^\top$ |
| | S4 [23] | LU *(tensor contraction)*<br>$\mathbf{L}_{ij} = \mathbf{C}^\top \mathbf{A}^{i-j} \mathbf{B}$ | $\mathbf{h}_i = \mathbf{A}\mathbf{h}_{i-1} + \mathbf{B}\mathbf{u}_i$<br>$\mathbf{y}_i = \mathbf{C}^\top \mathbf{h}_i + \mathbf{D}\mathbf{u}_i$ |

Table from 2026 survey paper [7].

# Attention: Motivation

© 2026 Fessler

Consider translating these two sentences to another language:

- Sentence 1: The trophy didn't fit into the brown suitcase because **it** was too large.
- Sentence 2: The trophy didn't fit into the brown suitcase because **it** was too small.
- In computational linguistics, identifying what "it" means (or what it "attends" to) is called **coreference resolution**.
- General **attention** ideas have been in the literature for decades.
- The **self-attention** mechanism, published in 2017 [9] as part of the Transformer architecture, was a breakthrough in NLP.
- Excellent overview in [1, §15.4].
- Useful for NLP and more, including images, *e.g.*, [8].
- Key idea:

**Transformer** architecture:

- Input is a **sequence** of vectors $\boldsymbol{x}_i$
- Output is a **sequence** of vectors $\boldsymbol{y}_i$
- **Self-attention** models interactions between vectors
- **Layer norm** and MLP work independently per vector
- Feedforward network with skip connections

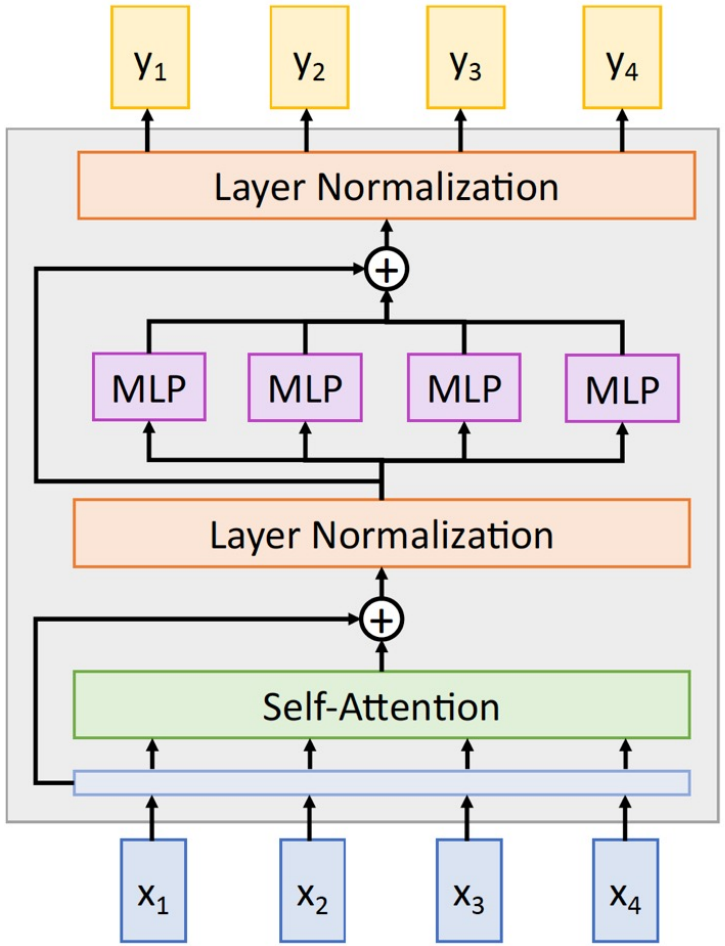



Figure courtesy [9].

## Self-attention

- Each position in a length-$N$ sequence has an embedded token $\boldsymbol{x} \in \mathbb{R}^d$ (image patch, tokenized word).
- We use learned weights to transform them to vectors $\boldsymbol{q}, \boldsymbol{k} \in \mathbb{R}^d$.
- These vectors form the rows of matrices $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{N \times d}$.
- Terminology from **databases**:
  - Query
  - Key
  - Value

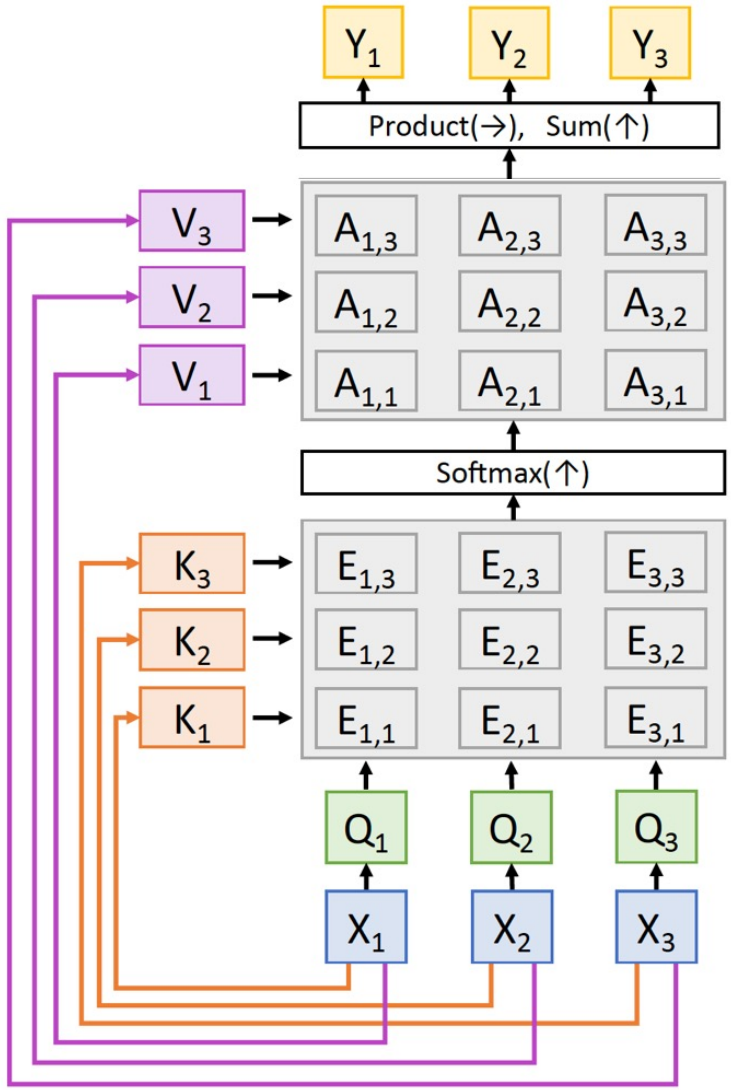



Figure courtesy [9].

From input $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, **self-attention** uses weight matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d \times}$ to compute

- $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{N \times}$
- $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \in \mathbb{R}^{N \times}$
- $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \in \mathbb{R}^{N \times}$
- finally outputting

- softmax applied to rows

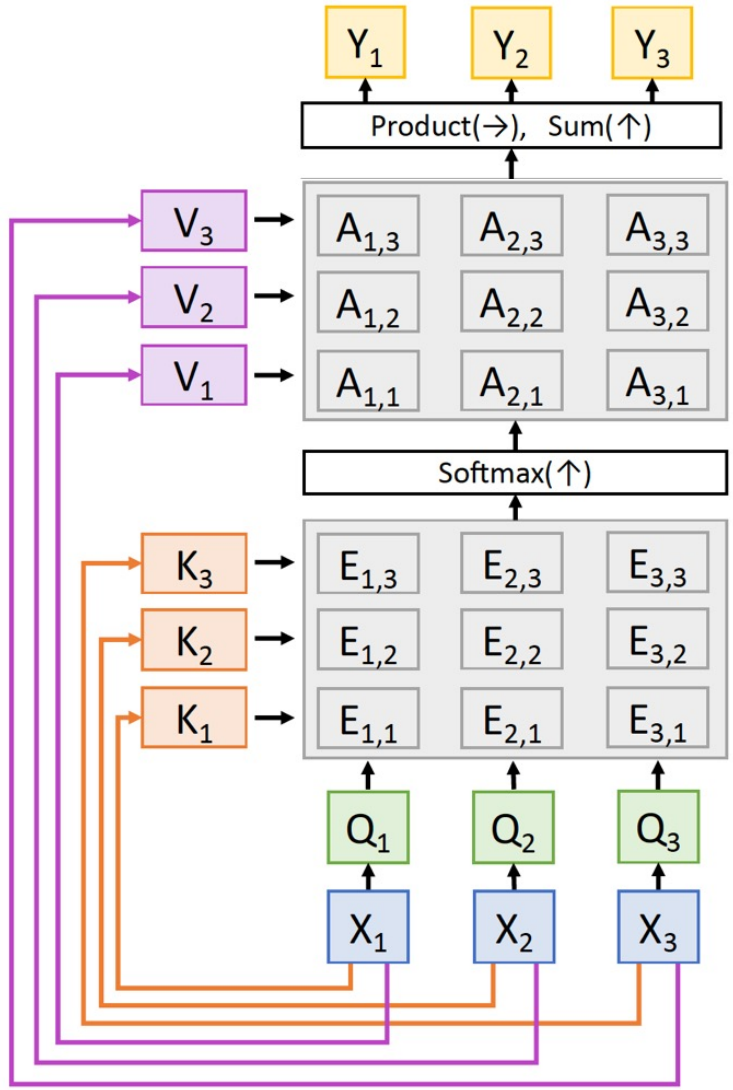



Figure courtesy [9].

# CQ: Attention computation

© 2026 Fessler

Q15.2 What is the computational complexity $O(\cdot)$ of forming $\boldsymbol{Q}\boldsymbol{K}^\top$ for $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{N \times d}$?

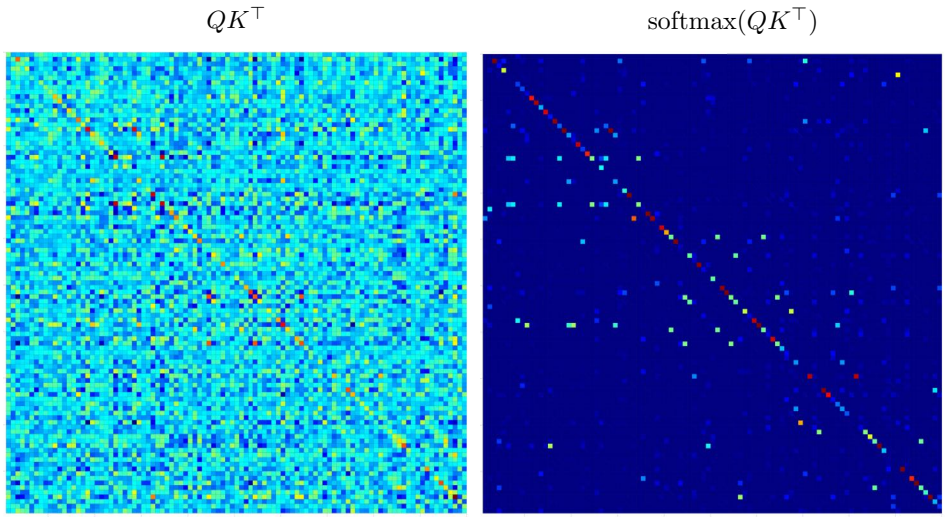
A: $N^3$ B: $N^2 d$ C: $N d^2$ D: $N^2$ E: none of these

Q15.3 A MLP applies a sequence of weight matrices $\{\boldsymbol{W}^{(l)}\}$, each followed by element-wise nonlinearities. A transformer block could be represented exactly by a suitably large (but finite) sequence of such matrices and (suitably chosen) element-wise nonlinearities ?

A: True B: False **??**

- The **softmax attention** matrix is defined as

$$\boldsymbol{A} = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)$$




- where softmax : $\mathbb{R}^N \mapsto \Delta^{N-1}$ (probability **simplex**) is applied row-wise:

$$[\text{softmax}(\boldsymbol{z})]_i = \frac{\mathrm{e}^{z_i}}{\sum_j \mathrm{e}^{z_j}}, \quad i = 1, \ldots, N.$$

# Many modern deep learning models...

© 2026 Fessler










text generation...




agentic AI...



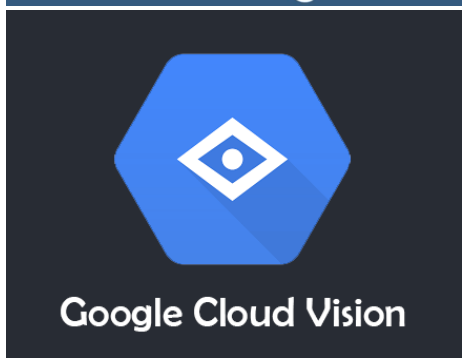




image recognition...







image generation...

# ...use transformer-based architectures

© 2026 Fessler

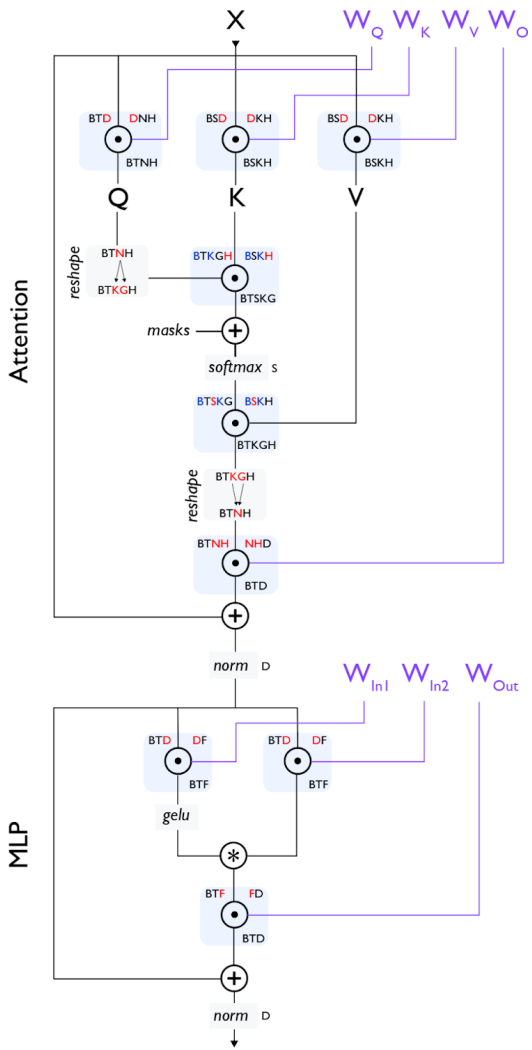



Figure courtesy [10].

- Each transformer block includes an attention layer and an MLP
- Attention involves the most computation and memory: generally $O(N^2 d)$ in terms of
  - **context length** $N$
  - **embedding dimension** $d$
  - GPT5 context $> 200k$ with 48 transformer blocks
  - Claude Sonnet 4 context $> 1M$ with unknown # blocks
  - Llama 3.1 context $> 128k$ with 126 transformer blocks
- Both context length and attention seem to be important for performance, but they are in tension for scaling.
- Some newer models replace MLP with mixture of experts [1, §13.6.2], using routing to decide with token goes to which MLP, so each MLP can be smaller.
- Chat bots use

# References I


[1] K. P. Murphy.
*Probabilistic machine learning: An introduction*.
MIT Press, 2022.

[2] SangGyu An.
Introduction to how an multilayer perceptron works but without complicated math, 2021.
https://medium.com/codex/introduction-to-how-an-multilayer-perceptron-works-but-without-complicated-math-a423979897ac.

[3] L. R. Rabiner.
An tutorial on hidden Markov models and selected applications in speech recognition.
*Proc. IEEE*, 77(2):257–86, February 1989.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton.
A simple framework for contrastive learning of visual representations.
In *Proc. Intl. Conf. Mach. Learn*, volume 119, pages 1597–607, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton.
A simple framework for contrastive learning of visual representations.
In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[6] O. Hsu and K. Koul.
Better hardware could turn zeros into AI heroes: Sparse computing enables leaner, faster AI.
*IEEE Spectrum*, 63(6):28–33, 2026.

[7] F. Sarnthein, A. Afzal, R. Pascanu, A. Orvieto, C. Gulcehre, J. Geiping, F. Tonin, A. Gu, and V. Cevher.
Sequence modeling architectures: Foundations.
*IEEE Sig. Proc. Mag.*, 43(3):115–29, 2026.

[8] H. Ramadan, D. E. Bourakadi, A. Yahyaouy, and H. Tairi.
Medical image registration in the era of Transformers: A recent review.
*Informatics in Medicine Unlocked*, 49:101540, 2024.

# References II


[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
*Advances in neural information processing systems*, 30, 2017.

[10] Jacob Austin, Sholto Douglas, Roy Frostig, Anselm Levskaya, Charlie Chen, Sharad Vikram, Federico Lebron, Peter Choy, Vinay Ramasesh, Albert Webson, and Reiner Pope.
*How to Scale Your Model*.
Google DeepMind, 2025.
Retrieved from https://jax-ml.github.io/scaling-book/.