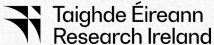
Meet xNNs

Feedforward neural networks (FNNs) and convolutional neural networks (CNNs)

Tobias Rossmann

Ollscoil na Gaillimhe
University of Galway

MathLearn — June 2026



Directed acyclic graphs (DAGs)

- A **directed graph** is a pair $G = (V, E)$, where V is a set of **vertices** and $E \subset V \times V$ is a set of **directed edges** between vertices.
- Let $v \in V$. Define

$$\mathrm{pred}(v) = \{u \in V : (u, v) \in E\}$$

- A **source** is a vertex which is not the tail of an edge. A **sink** is a vertex which is not the head of an edge.
- A **directed acyclic graph (DAG)** is a directed graph without oriented cycles. This just means that travelling along the directions of edges, you'll never run into a loop.
- A **feedforward neural network (FNN)** is a generalisation of MLPs built on DAGs.

Feedforward neural networks (FNNs)

- Let $G = (V, E)$ be a DAG. We refer to the vertices of G as the **units** or **neurons**.
- Each vertex $v \in V$ gives rise to a value $h_v \in \mathbf{R}$.
- Sources are inputs: for each source v, we are simply given $x_v = h_v \in \mathbf{R}$.
- For a non-source v, the value h_v is computed in the form

$$h_v = \sigma_v \left(\sum_{u \in \mathrm{pred}(v)} w_{uv} h_u + b_v \right)$$

  for edge weights w_{uv} and a bias b_v, and an activation function σ_v.
- Sinks are outputs.
- *Warning.* The word "feature" has multiple meanings in ML. For an input feature $h_v = x_v$, the subsequent h_w are **learned features** or **internal features**.

Layered FNNs

- Suppose that V is partitioned into **layers** $V_0, \dots, V_L$ such that $E \subseteq \bigcup_{\ell=1}^{L} V_{\ell-1} \times V_\ell$.
- The computation at $v \in V_\ell$ therefore takes the form

$$h_v = \sigma_v \left(\sum_{u \in \mathrm{pred}(v)} w_{uv} h_u + b_v \right).$$

- Write $d_\ell = |V_\ell|$. If $\sigma_v = \sigma_{v'} = \sigma_\ell$ for all $v, v' \in V_\ell$, then the various h_v with $v \in V_\ell$ assemble into layer vectors

$$h_\ell = \sigma_\ell(W_\ell h_{\ell-1} + b_\ell),$$

  for $W_\ell \in \mathrm{M}_{d_\ell \times d_{\ell-1}}(\mathbf{R})$ and $b_\ell \in \mathbf{R}^{d_\ell}$.
- G is **fully connected between consecutive layers** if $E = \bigcup_{\ell=1}^{L} V_{\ell-1} \times V_\ell$.

  In this case, we (essentially) recover MLPs!

Convolutional neural networks: overview

- Suppose we're facing the binary classification problem of recognising images of cats.
- If you translate or rotate an image of a cat, it's still an image of a cat.
- Mathematically speaking, felinity is invariant under the action of a group.
- *Convolutional neural networks (CNNs)* are neural networks that encode symmetry.

  Well, standard CNNs only encode translational symmetries, but the mathematics can do more.
- Their invention was inspired by the way that our visual cortex processes data. CNNs are the foundation of many advances in computer vision, image recognition, etc.

Convolutional neural networks: overview

- CNNs are designed for problems where the network should learn to detect patterns that are "local" in the input (e.g. recognise a cat, detect a singularity).
- It would be desirable if a learned detector for local patterns should also be usable in other locations. This is done via *parameter sharing*.
- Having fewer parameters can make learning easier and reduce overfitting.
- This lecture provides a high-level overview of what CNNs are and how they work from a mathematical perspective.

Images, tensors, vectors, and signals

- A greyscale image with a height of H pixels and a width of W pixels can be represented by an array in $\mathbf{R}^{\mathrm{H} \times W}$.
- Colour images are stored by fixing a number C of **channels** and storing a greyscale image for each channel.
- The **RGB colour model** has three channels: red, green, and blue.
- A colour image is a multidimensional array (or tensor) $\mathrm{x} \in \mathbf{R}^{\mathrm{C} \times \mathrm{H} \times W}$.
- If we wanted to pass x to an MLP, we'd "flatten" x into a vector $\mathrm{flatten}(\mathrm{x}) \in \mathbf{R}^{\mathrm{CH}W}$ and apply an affine linear transformation followed by an activation function.
- This ignores all spatial information stored in the array x! For instance, neighbouring pixels are geometrically close.
- CNNs are designed to capture this kind of geometric structure.

Review of group actions

- Let G be a group. Let X be a set.
- Recall that a **(left) action** of G on X is a function

$$G \times X \to X, \quad (g, x) \mapsto g.x = gx$$

  such that $1.x = x$ and $g.(h.x) = (gh).x$ for all $g, h \in G$ and $x \in X$.
- Every group G acts on itself via the **regular action** $g.h = gh$.
- We call a set X endowed with an action by G a G-**set**.

Equivariance and invariance

- Let X and Y be G-sets and let Z be a set.
- A function $T\colon X \to Y$ is G-**equivariant** if

$$T(g.x) = g.T(x)$$

  for all $g \in G$ and $x \in X$.
- A function $I\colon X \to Z$ is G-**invariant** if

$$I(g.x) = I(x)$$

  for all $g \in G$ and $x \in X$.
- Equivariance and invariance are both useful in ML:
  - Equivariance is useful for feature extraction (e.g. position of a cat).
  - Invariance is useful for classification (e.g. felinity).

Equivariance and invariance

- Let V be a vector space.
- Let $T\colon X \to V$ be any function. Then

$$I(x) := \sum_{g \in G} T(g.x)$$

is G-invariant (if the summation makes sense).

Images and signals

- A **signal** on G is a function $f\colon G \to \mathbf{R}$.
- G acts on signals via

$$(g.f)(x) = f(g^{-1}x).$$

- An image is a function $\{1,\dots,H\} \times \{1,\dots,W\} \to V$, where V is a vector space. Channels are coordinates on V.
- It's mathematically cleaner to extend this to a function $\mathbf{Z}^2 \to V$ which vanishes outside of the original grid.
- This turns the channels of images into signals on the additive group $\mathbf{Z}^2$.
- The natural translation action of $\mathbf{Z}^2$ on signals takes the form

$$(a.f)(x) = f(x-a).$$

Linear + equivariant = convolution

- Suppose that G is finite. The **convolution** $f_1 * f_2$ of $f_1, f_2 \colon G \to \mathbf{R}$ is defined by

$$(f_1 * f_2)(x) = \sum_{g \in G} f_1(g) f_2(g^{-1}x) = \sum_{g,h \in G \text{ with } gh = x} f_1(g) f_2(h).$$

Theorem

Let $T \colon \mathbf{R}^G \to \mathbf{R}^G$ be a G-equivariant linear map.
*Then there exists a unique **kernel** $k \in \mathbf{R}^G$ with $T(f) = f * k$ for all $f \in \mathbf{R}^G$.*

- Let G be arbitrary. For infinite G such as $\mathbf{Z}^2$, the above needs some tweaking.
- Let U and V be finite-dimensional vector spaces.
  Let $U^{(G)}$ be the space of *finitely supported* functions $G \to U$.
- Any G-equivariant linear map $T \colon U^{(G)} \to V^{(G)}$ is of the form $T(f) = f * k$ for a unique (finitely supported) kernel $k \colon G \to \mathrm{Hom}_{\mathbf{R}}(U, V)$.
- This kernel just assigns a matrix to each element of G.
  When $G = \mathbf{Z}^2$, the kernel is a 4-dimensional array of numbers.

You've seen convolution!

- ChatGPT informed me that a lecture on CNNs without pictures is a no go.
- Take an image. Let's reduce dimensions by first converting it to greyscale. Our image is then a signal $\mathbf{Z}^2 \to \mathbf{R}$.
- In the scalar case, the kernel of our convolution is just a (small) matrix.
- When applied to a single pixel, we take a weighted sum of the neighbouring pixels with coefficients taken from our matrix.
- The matrix then slides through our picture to transform it.

You've seen convolution!

- For example, the 8×8 filter

$$\frac{1}{64}\begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}$$

  blurs an image: the brightness of each pixel is replaced by the average brightness over all 64 points around it.
- This is called **box blur**. You may need run through a few passes to see the effect. Let's start with this image.

You've seen convolution!

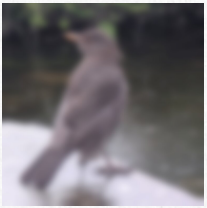
(a) 1 pass

(b) 2 passes

(c) 4 passes

(d) 8 passes

(e) 16 passes



(f) 32 passes

You've seen convolution!

- Filters can do many other useful things. For example, the kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

  can be used for edge detection.

CNNs

- The input layer of a CNN handles $h_0 \in V_0^{(G)}$.
  For example, we could have $G = \mathbf{Z}^2$ and $V_0 = \mathbf{R}^3$, in which case we are storing a 3-channel image.
- **Convolutional layers** are the main learned layers of CNNs.
  - They first map $h_{\ell-1} \in V_{\ell-1}^{(G)}$ to

$$z_\ell = h_{\ell-1} * k_\ell + b_\ell \in V_\ell^{(G)}$$

    for a kernel or **filter** $k_\ell \in \mathrm{Hom}_{\mathbf{R}}(V_{\ell-1}, V_\ell)^{(G)}$ and (shared) bias $b_\ell \in V_\ell$.
  - Typically, K_ℓ will have small support which enforces *locality*. The support could be centred around the identity or spaced out by means of a *stride* parameter.
  - Through convolution, the kernel k_ℓ is used across G, which realises *weight sharing*.
  - As always, an activation function (e.g. ReLU) is applied to z_ℓ to produce $h_\ell = \sigma_\ell(z_\ell)$.

CNNs

- A **global pooling** / **averaging layer** sends $f \in V_\ell^{(G)}$ to $\sum_{g \in G} f(g)$.

  This turns equivariant features into invariant ones.

  Intuition: instead of tracking where something occurs, we measure how strongly it occurs overall.
- A final fully connected / dense affine layer. This might e.g. be used to convert learned features to logits.
- A final activation as always (e.g. softmax for classification).

CNNs in practice

- For $G = \mathbf{Z}^2$, we store images as multiarrays from $\mathbf{R}^{C_\ell \times H_\ell \times W_\ell}$.
- Convolution kernels are stored as multiarrays in $\mathbf{R}^{C_\ell \times C_{\ell-1} \times R_\ell \times S_\ell}$. This corresponds to a function $\mathbf{Z}^2 \to M_{C_\ell \times C_{\ell-1}}(\mathbf{R})$ which is supported on $\{1, \dots, R_\ell\} \times \{1, \dots, S_\ell\}$.
- To control output size, one often pads the input, commonly with zeros. Boundary handling is a practical detail.
- ML libraries often don't use (mathematical) convolution

$$(f * k)(x) = \sum_{g \in G} k(g^{-1}x)(f(g))$$

but **cross-correlation**

$$(f \star k)(x) = \sum_{g \in G} k(gx)(f(g)).$$

In the ML literature, cross-correlation is also referred to as convolution.

Lessons to take away from this lecture

- Vanilla MLPs ignore geometry.
- CNNs impose locality and parameter sharing.
- Convolution is the mathematical operation behind linear equivariant maps. There's algebra in machine learning!