

Keras API for TensorFlow

Developing Artificial & Convolutional Neural Networks with Python's Keras API

Ryan Paul Lafler

Samuel S.P. Shen

Mitch Goldberg

AMS 2025

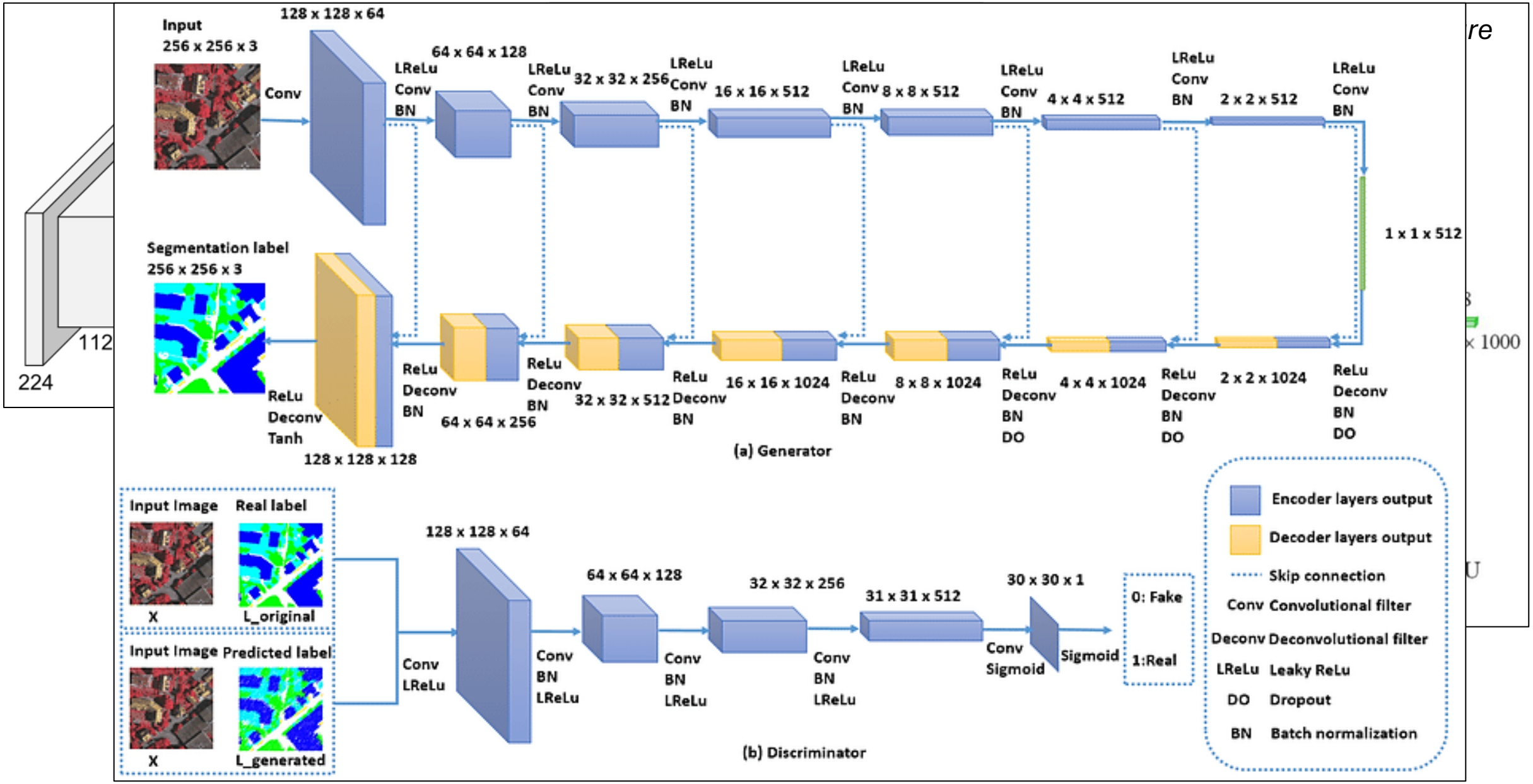
105TH ANNUAL MEETING

12–16 JANUARY

NEW ORLEANS & ONLINE

AMERICAN METEOROLOGICAL SOCIETY

The Vast, Never-Ending World of Deep Learning Architectures



Deep Learning (DL) for Gridded Climate Datasets

- Deep Learning (DL) infrastructures accept, generate, and naturally work with gridded, array-based datasets (tensors) to automate, predict, extract, forecast, & generate result
- **Spatial Resolution Downscaling (Upsampling)** → Enhance coarse grids to show finer, localized details at a higher resolution
- **Temporal Resolution Upsampling** → Increase frequency of data reporting, measurements, & simulations by generating data to fill temporal gaps
- **Anomaly Detection** → Detect significant departures (anomalous events) from typical norms, implement with forecasting to understand extreme events
- **Event Detection** → Automate processes to identify, classify, & monitor events in real-time
- **Forecasting** → Use historical climate arrays to forecast global, regional, or local events for different variables several time-steps in advance

DL in Python: Libraries, Integrations, & APIs with Keras

- Python supports many libraries / frameworks for DL: **TensorFlow**, **Pytorch**,& **Theano**
- **Keras** is a high-level framework that allows researchers & scientists to focus on deploying DL architectures, rather than building everything from scratch → builds backends to DL libraries
  - Various APIs in Keras to accomplish this → **Sequential**, **Functional**, & **Subclassing**
  - **Sequential API** allows developers to create feed-forward Networks using stack of layers
  - **Functional API** allows developers to create multi-input, multi-output, & skip-layer Networks by building advanced architectures for wide variety of data
  - **Subclassing API** allows developers to create customized Network layers, objective (loss) functions, & architectures using Python classes with **`keras.Model`**

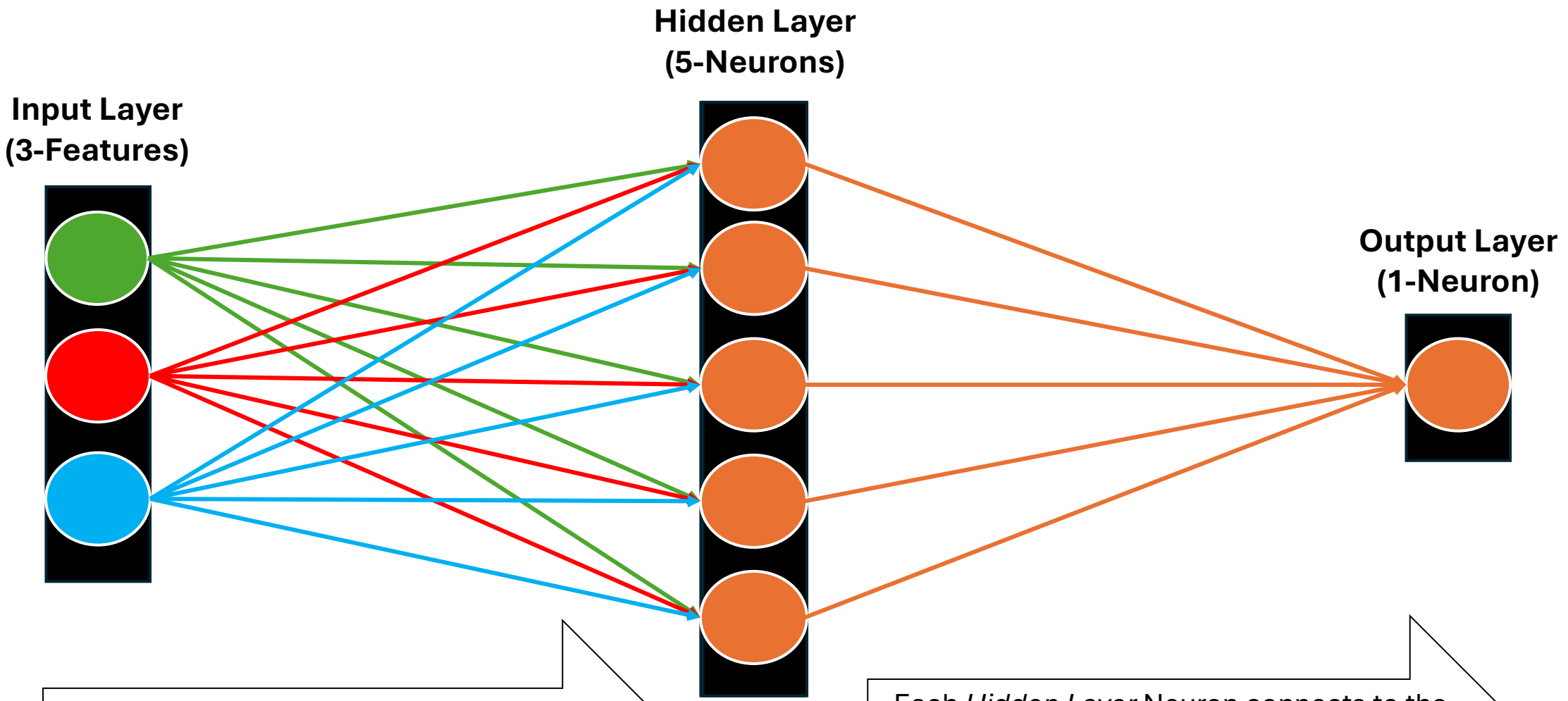


105TH ANNUAL MEETING
12–16 JANUARY
NEW ORLEANS & ONLINE
AMERICAN METEOROLOGICAL SOCIETY
AMS 2025

1. Artificial Neural Network (ANN) Fundamentals

Copyright © 2024-2025 by Ryan Paul Lafler.

Anatomy of a Fully-Connected Neural Network (ANN)



Each *Input Layer* Neuron connects to the 5-Neurons in the *Hidden Layer*.
$3 \times 5 = 15$ total connections.

Each *Hidden Layer* Neuron connects to the 1-Neuron in the *Output Layer*.
$5 \times 1 = 5$ total connections.

Artificial Neural Network Fundamentals

- Neurons from previous layer connected to neurons in current layer
  - Data only *feeds-forward* → does not skip layers or loop in repetition
- Deep Networks contain large stack of layers
- Wide Networks contains large numbers of neurons within layers
- Connections between layers are called **edges** that form **trainable weights** of Network
  - Connections form between neurons in adjacent layers
- **Activation functions** allow Networks to handle complex **non-linear** problems
  - **Non-linearity** allows Networks to be flexible, versatile, and entirely data-driven with minimal assumptions
  - Networks highly prone to overfitting → requires layers, dropout, & more to mitigate



2. Convolutional Neural Network (CNN) Fundamentals

Copyright © 2024-2025 by Ryan Paul Lafler.

Anatomy of a Convolutional Neural Network (CNN)

x_{11}	x_{12}	x_{13}
x_{21}	x_{22}	x_{23}
x_{31}	x_{32}	x_{33}

Input layer that contains original data values. Gridded data comes in a (3 x 3) array.

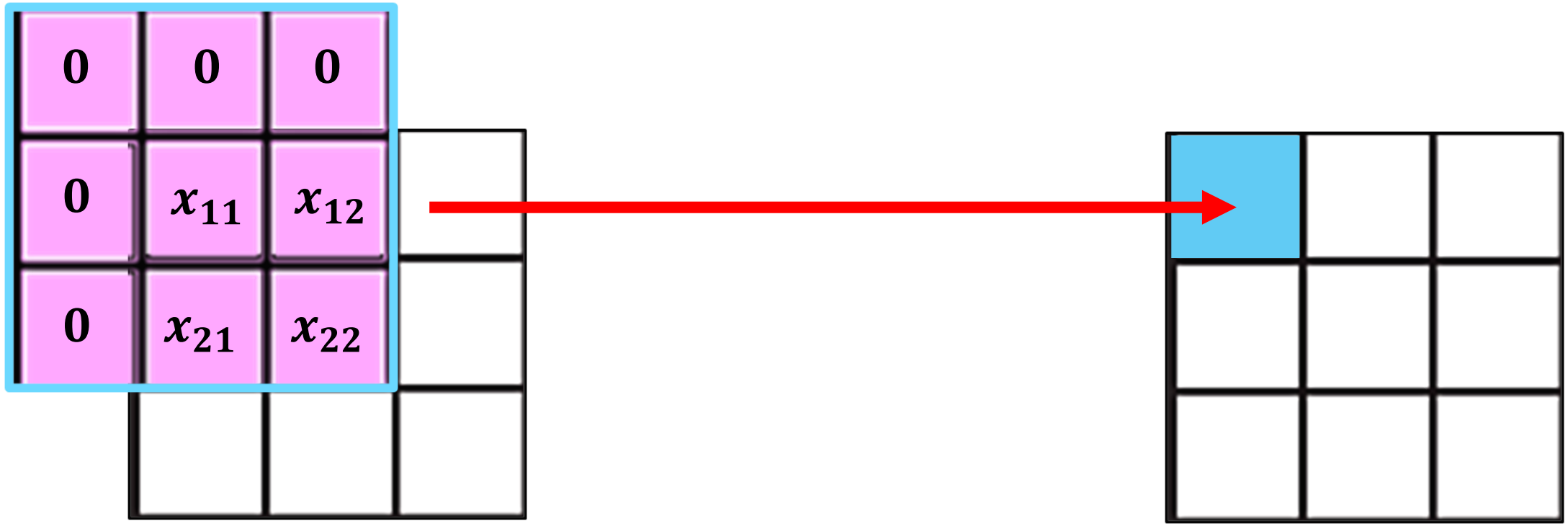
Center of kernel moves over every pixel, allowing spatial information to be preserved & used in Network.

-0.5	1.2	0.4
-1.4	0.7	-1.3
0.8	1.7	0.9

Kernel containing weights that are randomly initialized. Weights are shared between pixel values. CNN reduces number of trainable parameters by **weight sharing**.

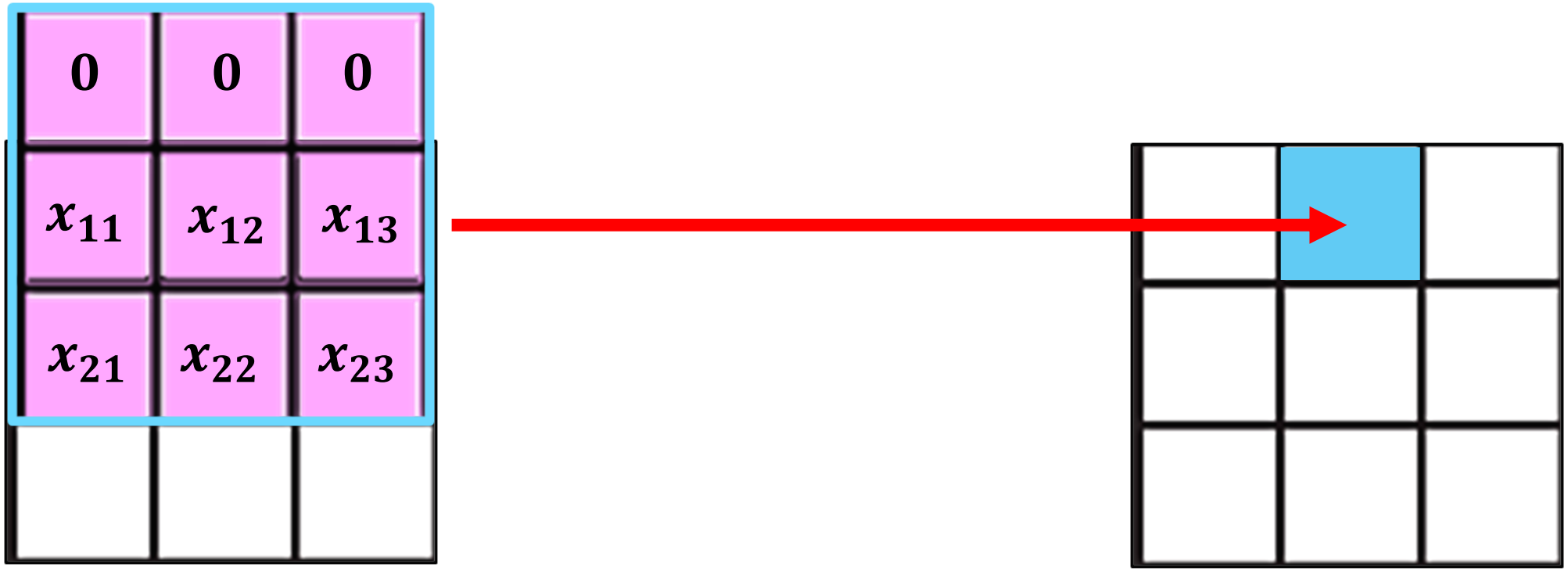
Weights are multiplied with respective pixel values.

Anatomy of a Convolutional Neural Network (CNN)



Kernel weights are randomly initialized from Uniform distribution. Nearby pixels captured by kernel are multiplied with weights to compute weighted sum and create target pixel.

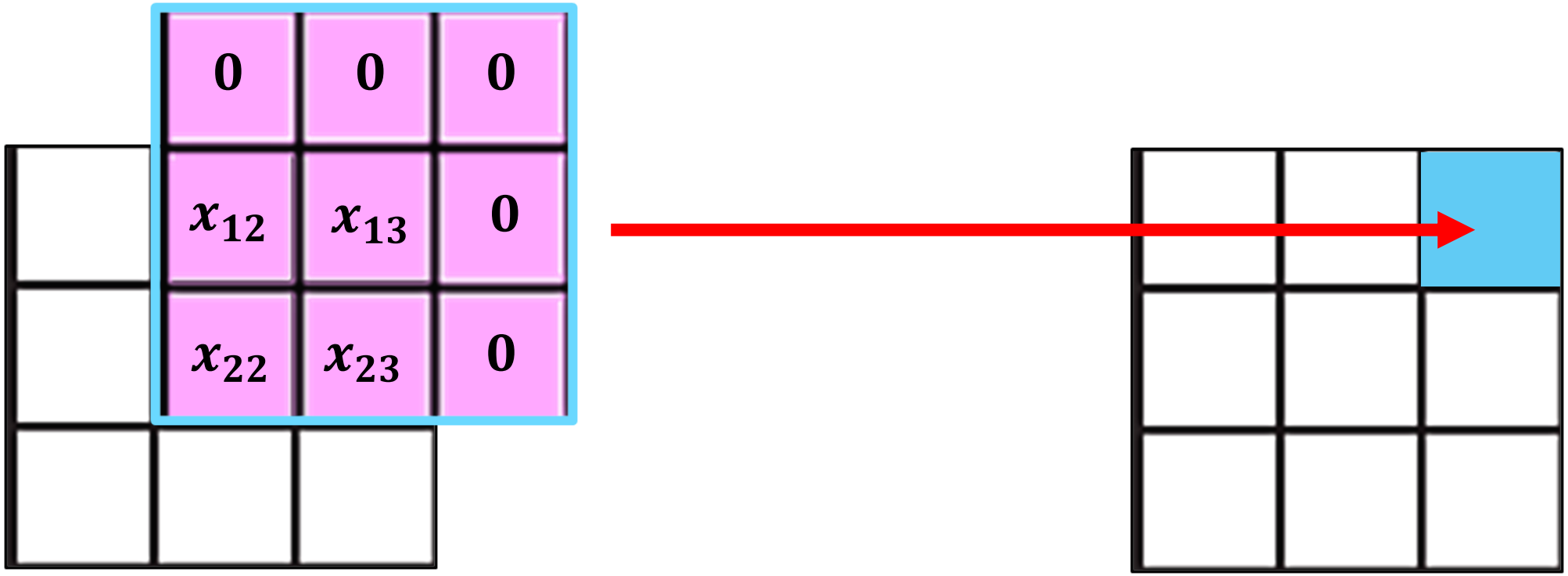
Anatomy of a Convolutional Neural Network (CNN)



Kernel uses zero-padding to ensure boundary pixels included when *scanning* gridded array. Preserves array's original dimensions; no information loss.

Kernel accounts for nearby relationships between pixels → pattern recognition.

Anatomy of a Convolutional Neural Network (CNN)



Kernel calculates weighted sum for all pixels inside of input array, looking at groupings instead of one pixel at-a-time like an ANN.

Factors in spatial associations & nearby relationships *shared* by pixels.

Anatomy of a Convolutional Neural Network (CNN)



Kernel calculates weighted sum for all pixels inside of input array, looking at groupings instead of one pixel at-a-time like an ANN.

Factors in spatial associations & nearby relationships *shared* by pixels.

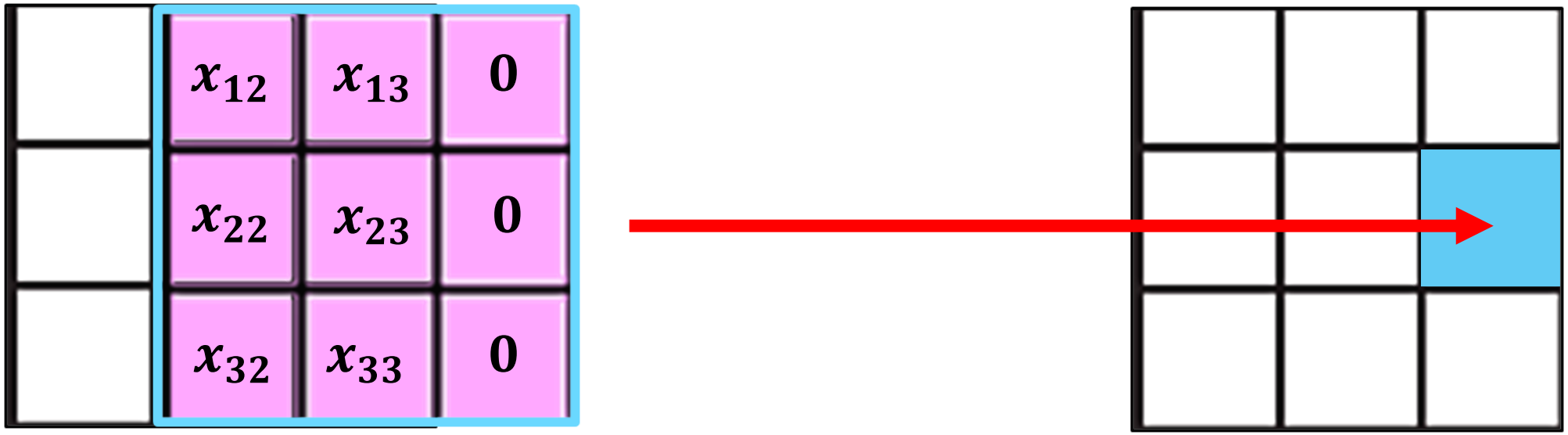
Anatomy of a Convolutional Neural Network (CNN)



Kernel calculates weighted sum for all pixels inside of input array, looking at groupings instead of one pixel at-a-time like an ANN.

Factors in spatial associations & nearby relationships *shared* by pixels.

Anatomy of a Convolutional Neural Network (CNN)



Kernel calculates weighted sum for all pixels inside of input array, looking at groupings instead of one pixel at-a-time like an ANN.

Factors in spatial associations & nearby relationships *shared* by pixels.

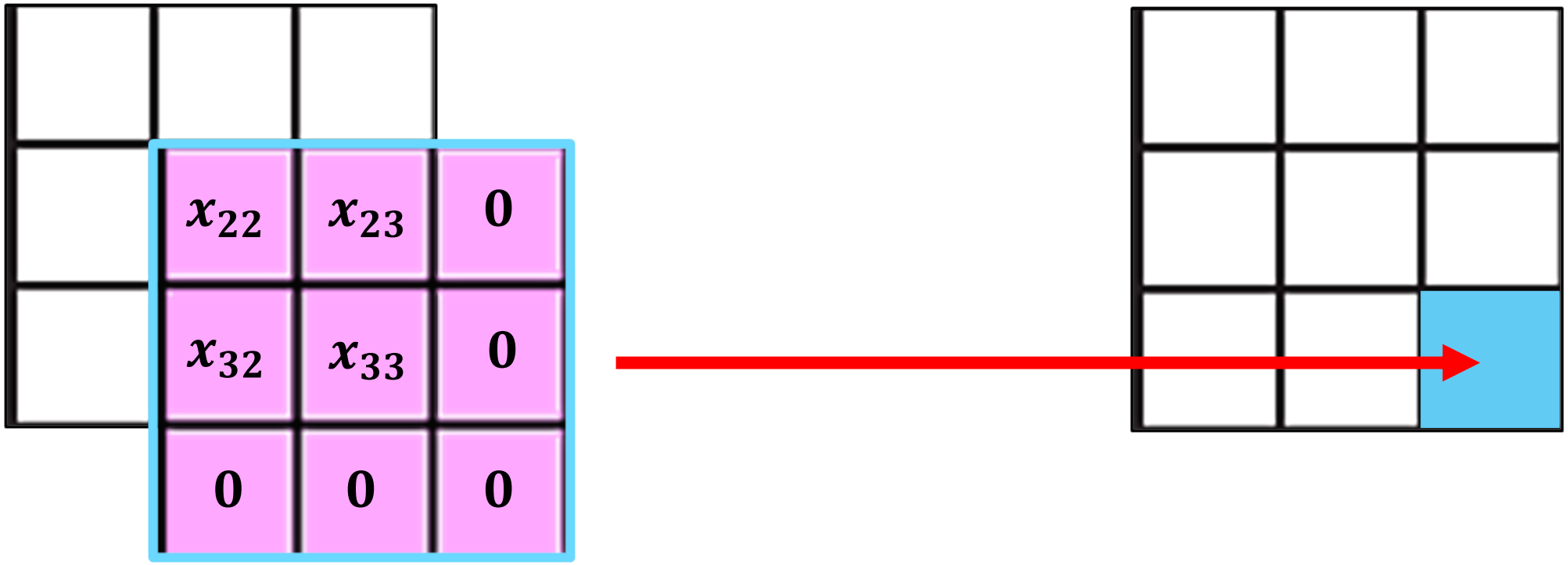
Anatomy of a Convolutional Neural Network (CNN)



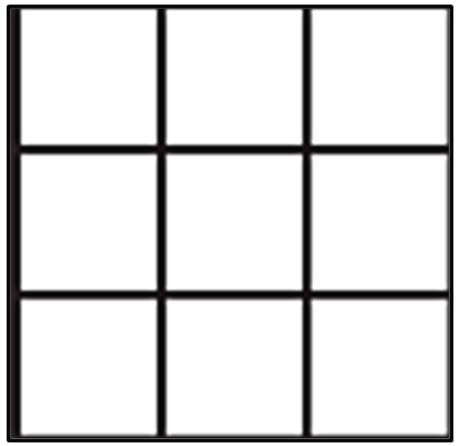
Anatomy of a Convolutional Neural Network (CNN)



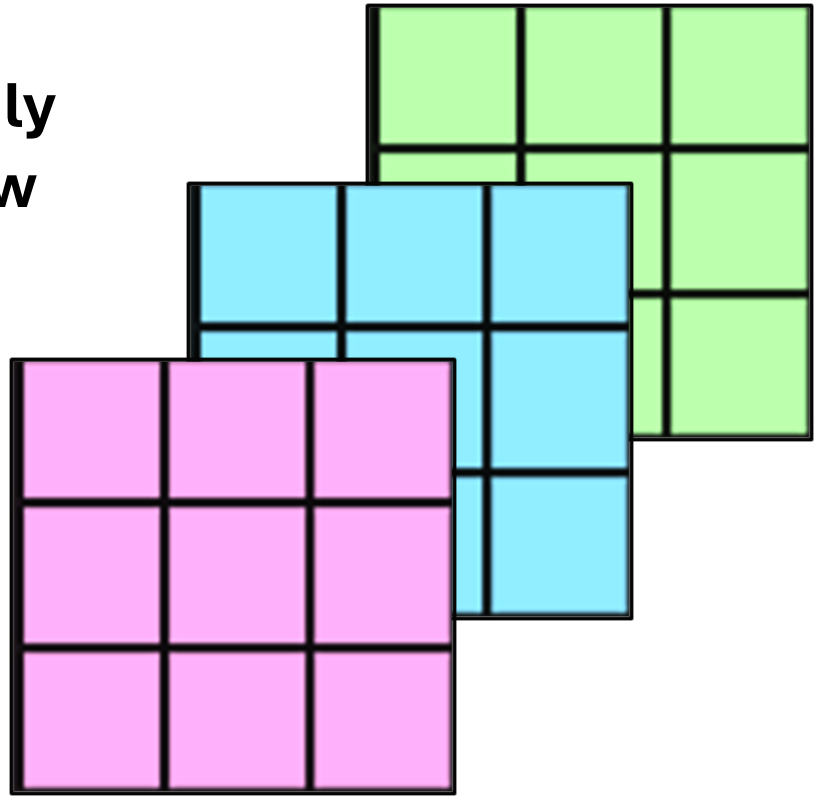
Anatomy of a Convolutional Neural Network (CNN)



Adding Channels (Feature Mappings) to a CNN



Repeat process using 3 randomly initialized kernels to make 3 new channels



Can produce multiple channels (feature mappings) by randomly initializing more than 1 kernel. To make 3 new channels, 3 kernels must examine groupings around each pixel. Kernels will always be different because of random initialization.



105TH ANNUAL MEETING
12–16 JANUARY
NEW ORLEANS & ONLINE
AMERICAN METEOROLOGICAL SOCIETY
AMS 2025

3. Building a CNN for Gridded Climate Data Using the Keras Functional API

Copyright © 2024-2025 by Ryan Paul Lafler.

Building a CNN Architecture for Gridded Climate Data Classification

- As we've seen, CNNs are excellent at learning patterns *and* preserving spatial relationships
- CNNs are also **translation invariant**: can recognize patterns *anywhere* in dataset, not limited to where they've seen them before
- Because of their versatility & natural handling of array-based data, let's develop a CNN using convolutional layers, pooling layers, and a densely connected Network to:
  - Input gridded climate arrays & output a probability for binary classification
  - Have a CNN detect & determine whether extreme weather events (or anomalies) are occurring at the current hour → assign a probability
    - **"1"** indicates anomalous
    - **"0"** indicates normal event
- Let's build the CNN using the Keras Functional API!

Creating a CNN for Gridded Data Classification



```
# Input layer
input_layer = Input(
    shape = (100, 100, 1),
    name = 'input'
)
# First convolutional process
x = Conv2D(
    filters = 3, kernel_size = (2, 2),
    padding = 'valid', activation = 'relu'
) (input_layer)
```

Input layer accepts (100 x 100) array with a single channel. 3-filters (kernels) of dimensions (2 x 2) are run over zero-padded array to generate 3-new channels.

Creating a CNN for Gridded Data Classification



```
# Second convolutional process
x = Conv2D(
    filters = 10, kernel_size = (3, 3),
    padding = 'valid',
    activation = 'relu'
) (x)
x = MaxPooling2D(
    (2, 2), padding = 'valid'
) (x)
```

Second convolutional process that creates 10 new channels using 10-kernels of size (3 x 3) moving across every pixel. Downscale resolution of image by max aggregation.

Creating a CNN for Gridded Data Classification



```
# Third convolutional process
x = Conv2D(
    filters = 15, kernel_size = (2, 2),
    padding = 'valid',
    activation = 'relu'
) (x)
x = MaxPooling2D(
    (2, 2), padding = 'valid'
) (x)
```

Third convolutional process to further increase feature mappings and reduce overall resolution of gridded data (provides new perspectives to view transformed data).

Creating a CNN for Gridded Data Classification



```
# Fourth convolutional process
x = Conv2D(
    filters = 20, kernel_size = (2, 2),
    padding = 'valid',
    activation = 'relu',
) (x)

# Flattening layer
x = Flatten() (x)
```

Final convolutional process to extract features from gridded data. Last part of convolutional process includes flattening layer for classification with ANN.

Creating a CNN for Gridded Data Classification



```
# Densely connected layers
x = Dense(
    64 , activation = 'relu'
) (x)
x = Dense(
    16 , activation = 'relu'
) (x)
x = Dense(
    8 , activation = 'relu'
) (x)
```

Feed-forward densely connected layers with ReLU activation to for binary gridded data classification tasks.

Creating a CNN for Gridded Data Classification

```
# Output layer
output_layer = Dense(
    1 ,
    activation = 'sigmoid'
) (x)
```

Output layer with a single neuron that utilizes **sigmoid** activation function for **binary classification** → good for modeling detection of weather disturbance / anomaly.

CNN Architecture for Binary Classification of Gridded Data



Model: "CNN"

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 150, 150, 1)	0
conv2d (Conv2D)	(None, 149, 149, 3)	15
conv2d_1 (Conv2D)	(None, 148, 148, 10)	130
max_pooling2d (MaxPooling2D)	(None, 74, 74, 10)	0
conv2d_2 (Conv2D)	(None, 73, 73, 15)	615
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 15)	0
conv2d_3 (Conv2D)	(None, 35, 35, 15)	915
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 15)	0
flatten (Flatten)	(None, 4335)	0
dense (Dense)	(None, 64)	277,504
dense_1 (Dense)	(None, 16)	1,040
dense_2 (Dense)	(None, 8)	136
dense_3 (Dense)	(None, 1)	9

Total params: 280,364 (1.07 MB)

Trainable params: 280,364 (1.07 MB)

Non-trainable params: 0 (0.00 B)



Thank You for Attending!

Questions, Comments, and/or Further Discussion?

Ryan Paul Lafler
rplafler@premier-analytics.com

Samuel S.P. Shen
sshen@sdsu.edu

Mitch Goldberg
mgoldberg1@ccny.cuny.edu

105TH ANNUAL MEETING
12–16 JANUARY
NEW ORLEANS & ONLINE
AMERICAN METEOROLOGICAL SOCIETY
AMS 2025