Computer Vision

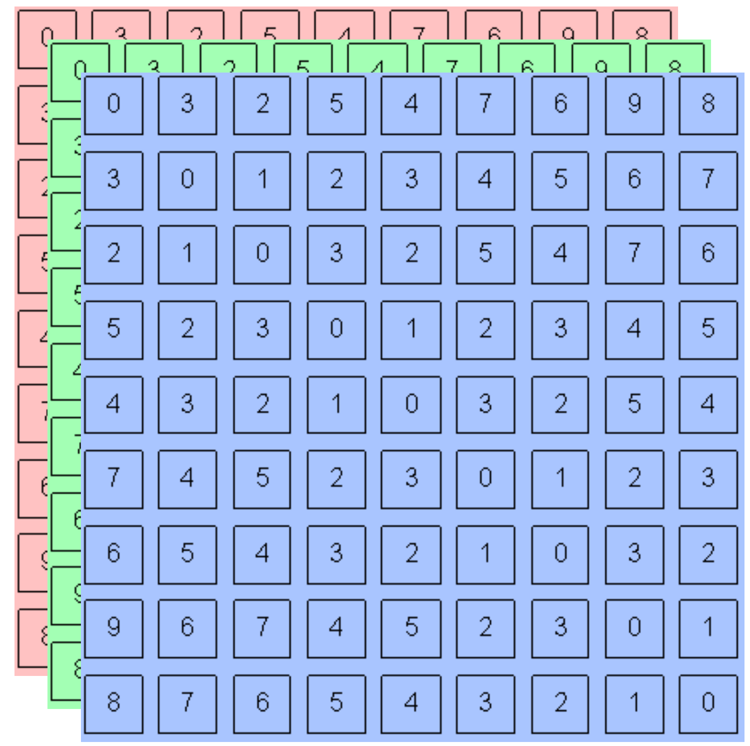
Slides adapted from Mohit Iyyer

grayscale images are matrices

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

what range of values can each pixel take?

color images are tensors



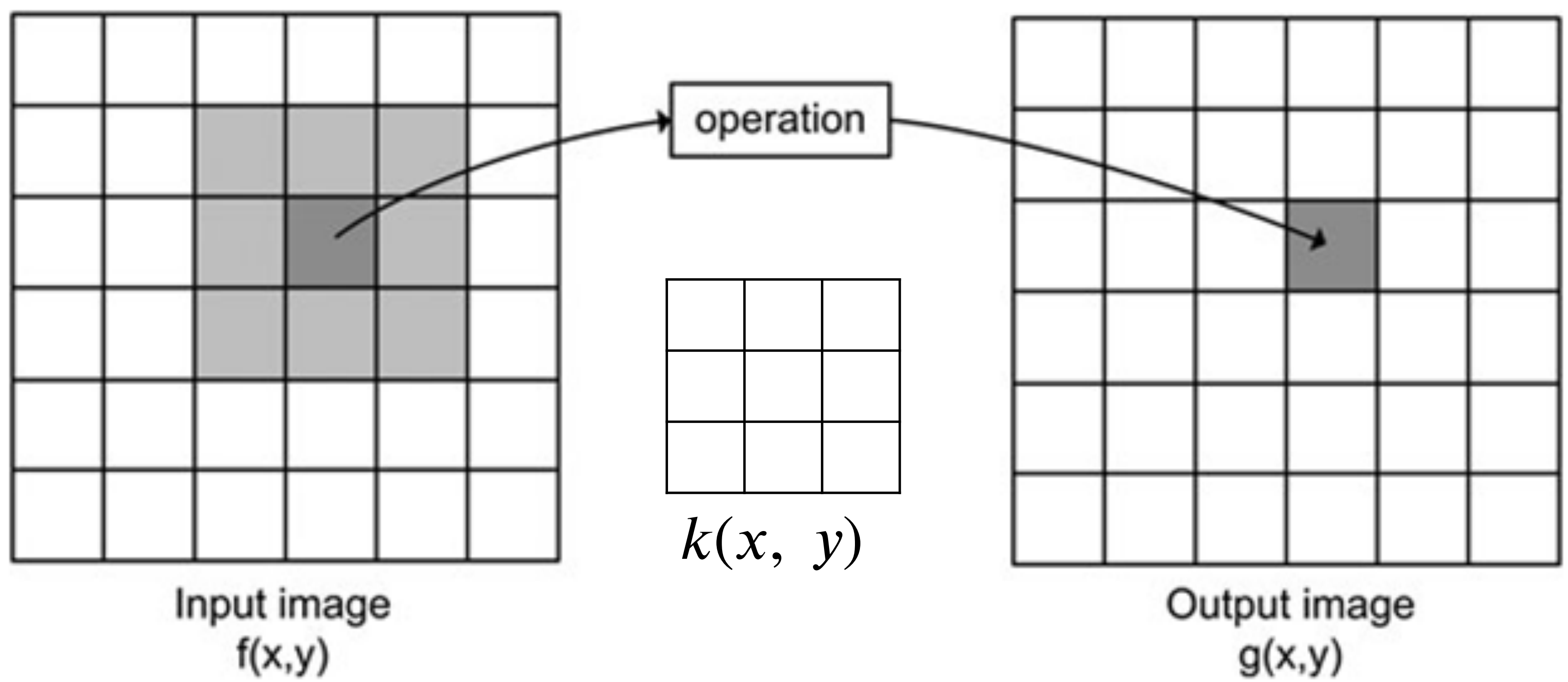
channel x height x width

Channels are usually RGB: Red, Green, and Blue

Other color spaces: HSV, HSL, LUV, XYZ, Lab, CMYK, etc

Convolutional Neural Networks

Convolution operator



$k(x,\ y)$

$$g(x,\ y) = \sum_{v} \sum_{u} k(u,v) f(x - u,\ y - v)$$

Image Credit: http://what-when-how.com/introduction-to-video-and-image-processing/neighborhood-processing-introduction-to-video-and-image-processing-part-1/

Slides adapted from Mohit Iyyer

(filter, kernel)

Input image * Weights Output image

4	5	7	6	6
3	2	8	0	7
6	7	7	1	5
3	0	1	1	1
4	3	2	1	7

*

0	0	0
1	0	1
0	0	0

→

	11	2	15	
	13	8	12	
	?			

Slides adapted from Mohit Iyyer

demo:
http://setosa.io/ev/image-kernels/

Slides adapted from Mohit Iyyer

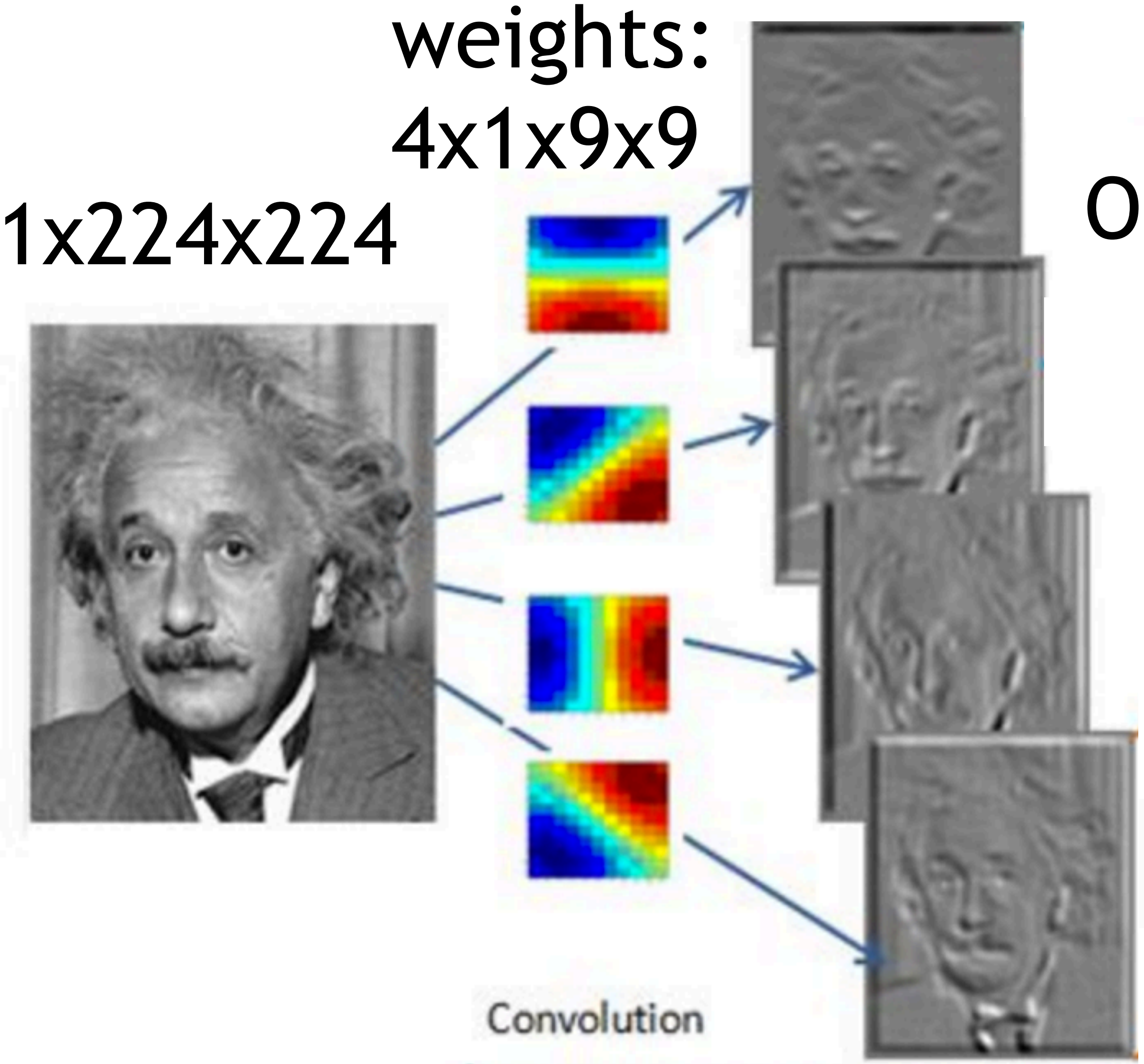
Convolutional Layer (with 4 filters)

weights:
4x1x9x9

Input: 1x224x224

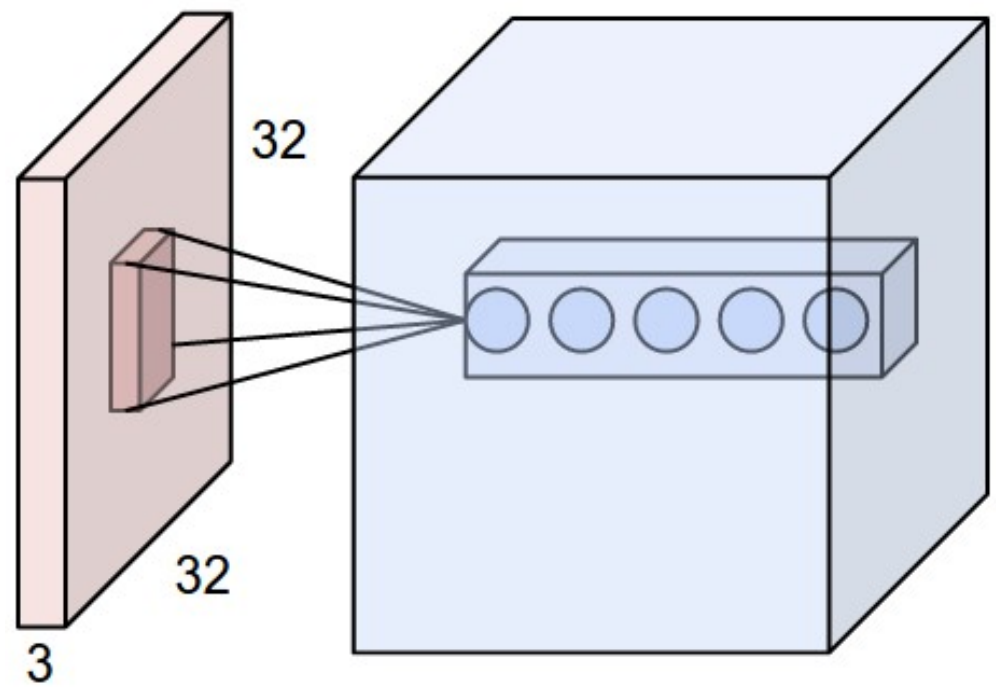
Output: 4x224x224

if zero padding,
and stride = 1



pooling layers also used to reduce dimensionality

Convolutional Layers:
slide a set of small filters over the image



Pooling Layers:
reduce dimensionality of representation

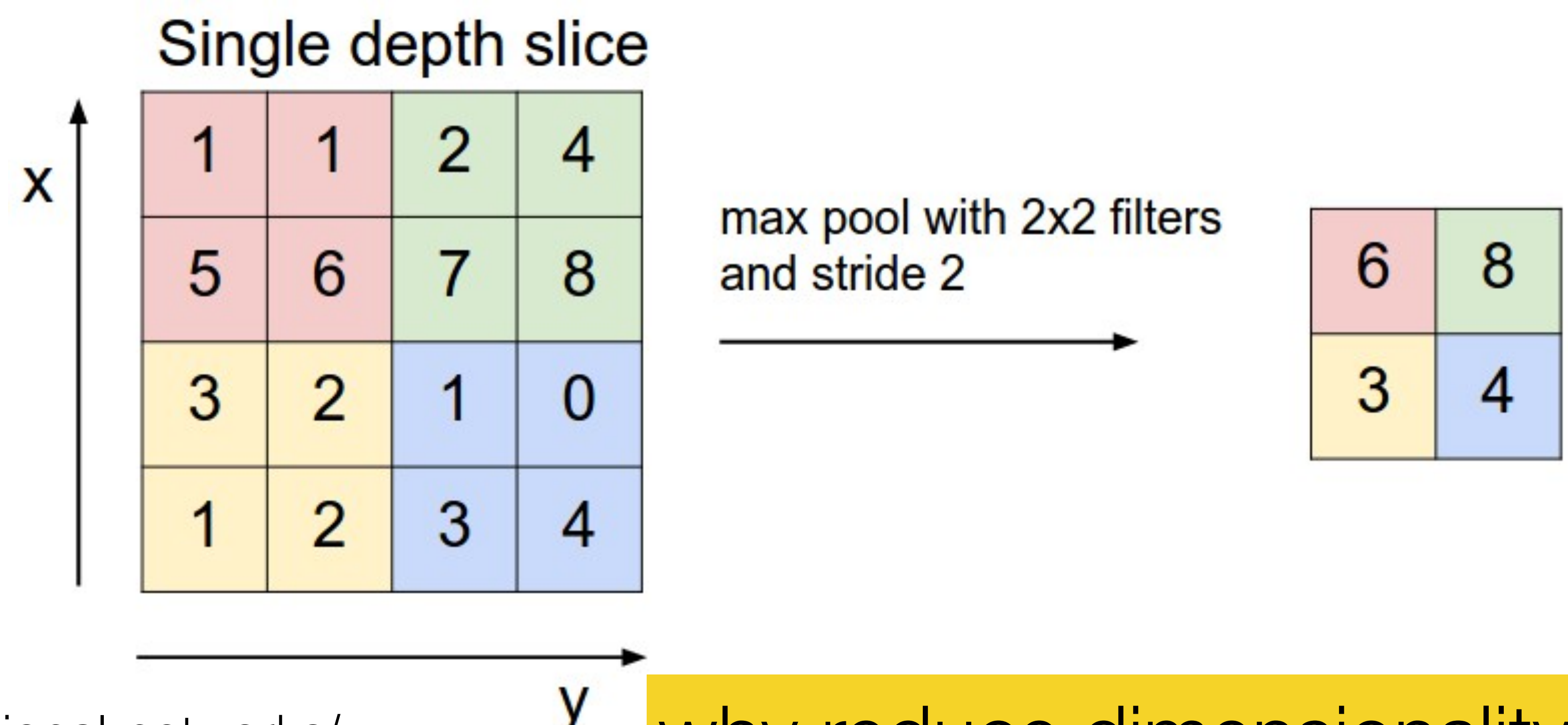


image: https://cs231n.github.io/convolutional-networks/

why reduce dimensionality?

Alexnet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

the paper that started the deep learning revolution!

image classification

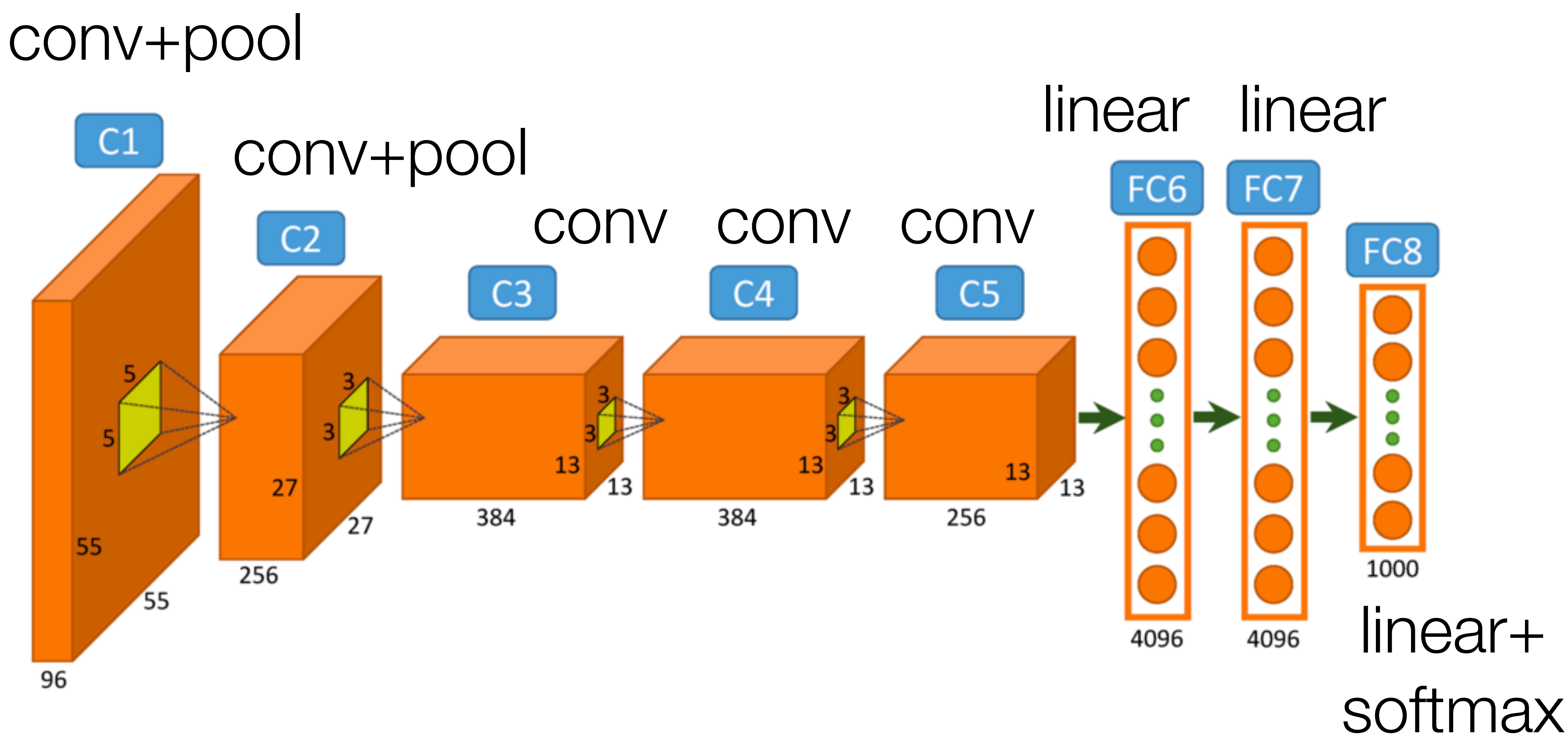
Classify an image into 1000 possible classes:
e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee,
red fox, banjo, barbell, hourglass, knot, maze, viaduct, etc.

cat, tabby cat (0.71)
Egyptian cat (0.22)
red fox (0.11)
.....

train on the ImageNet challenge dataset, ~1.2 million images

Slides adapted from Mohit Iyyer

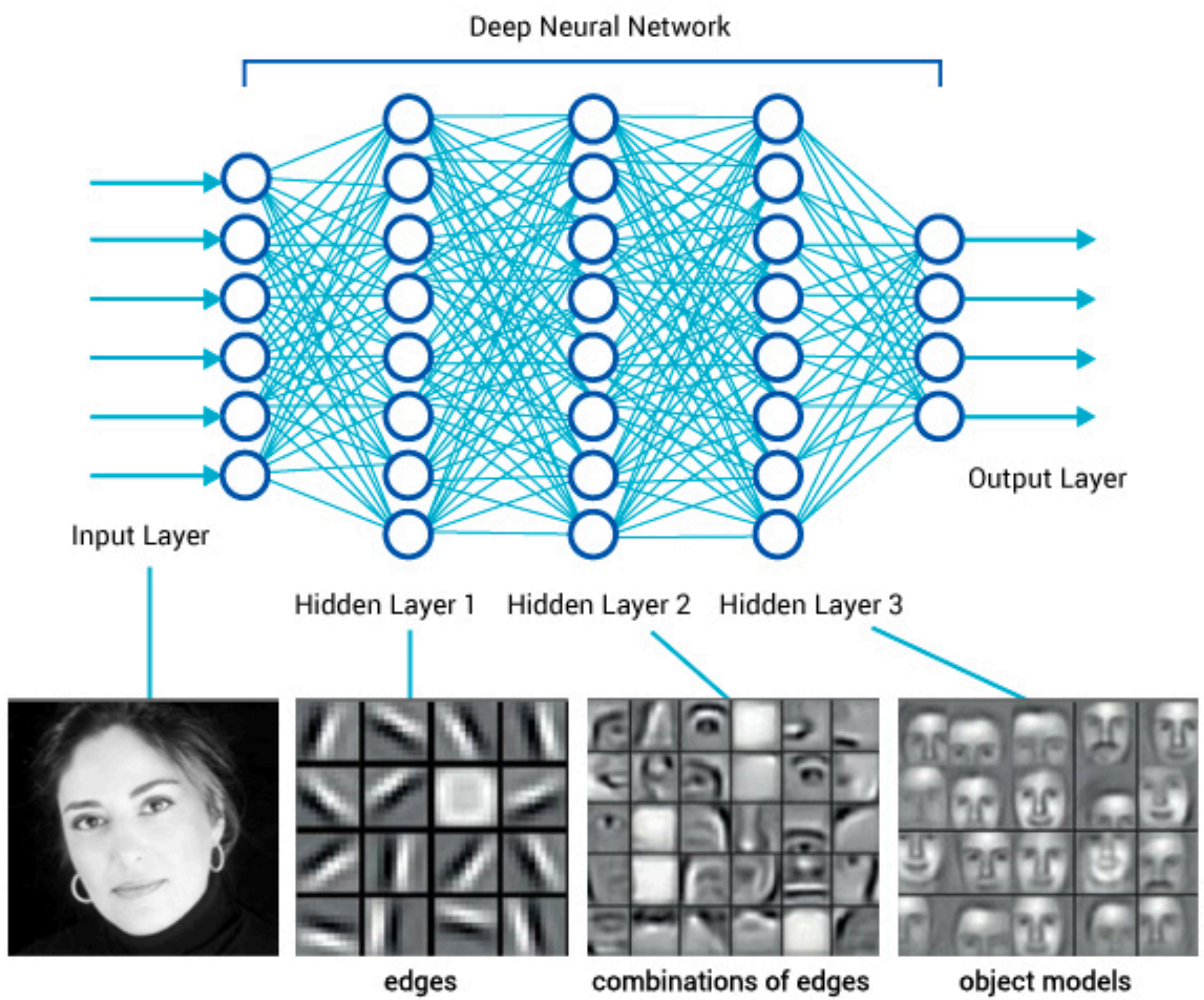
Alexnet



https://www.saagie.com/fr/blog/object-detection-part1

Slides adapted from Mohit Iyyer

What is happening?



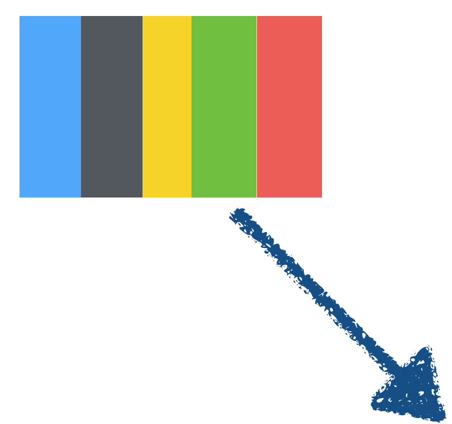
https://www.saagie.com/fr/blog/object-detection-part1

Slides adapted from Mohit Iyyer

at the end of the day, we generate a fixed size vector from an image and run a classifier over it

CNN (

) =

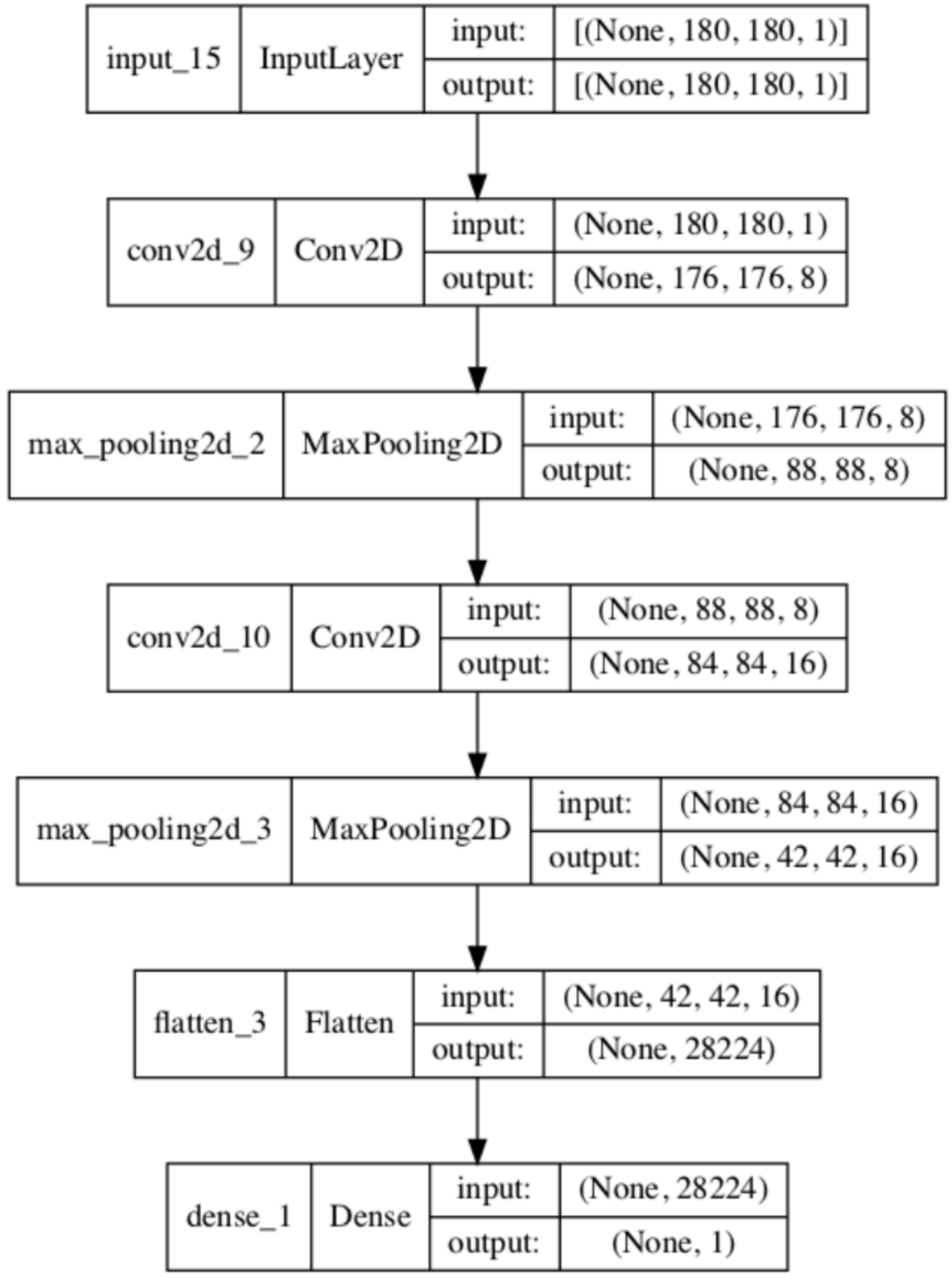


softmax: predict ‘truck’

Adding More Layers

```
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    x = layers.Conv2D(8, (5, 5), activation='relu', strides=1)(inputs)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(16, (5, 5), activation='relu', strides=1)(x)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Flatten()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)


model = make_model(input_shape=image_size+(1,), num_classes=2)
keras.utils.plot_model(model, show_shapes=True)
```

New Architecture



Auto-Encoders

Auto-encoders

Auto-encoders are a class of neural networks that do not require labeled data.

Supervised NNs: predict the **output** given the **input**.

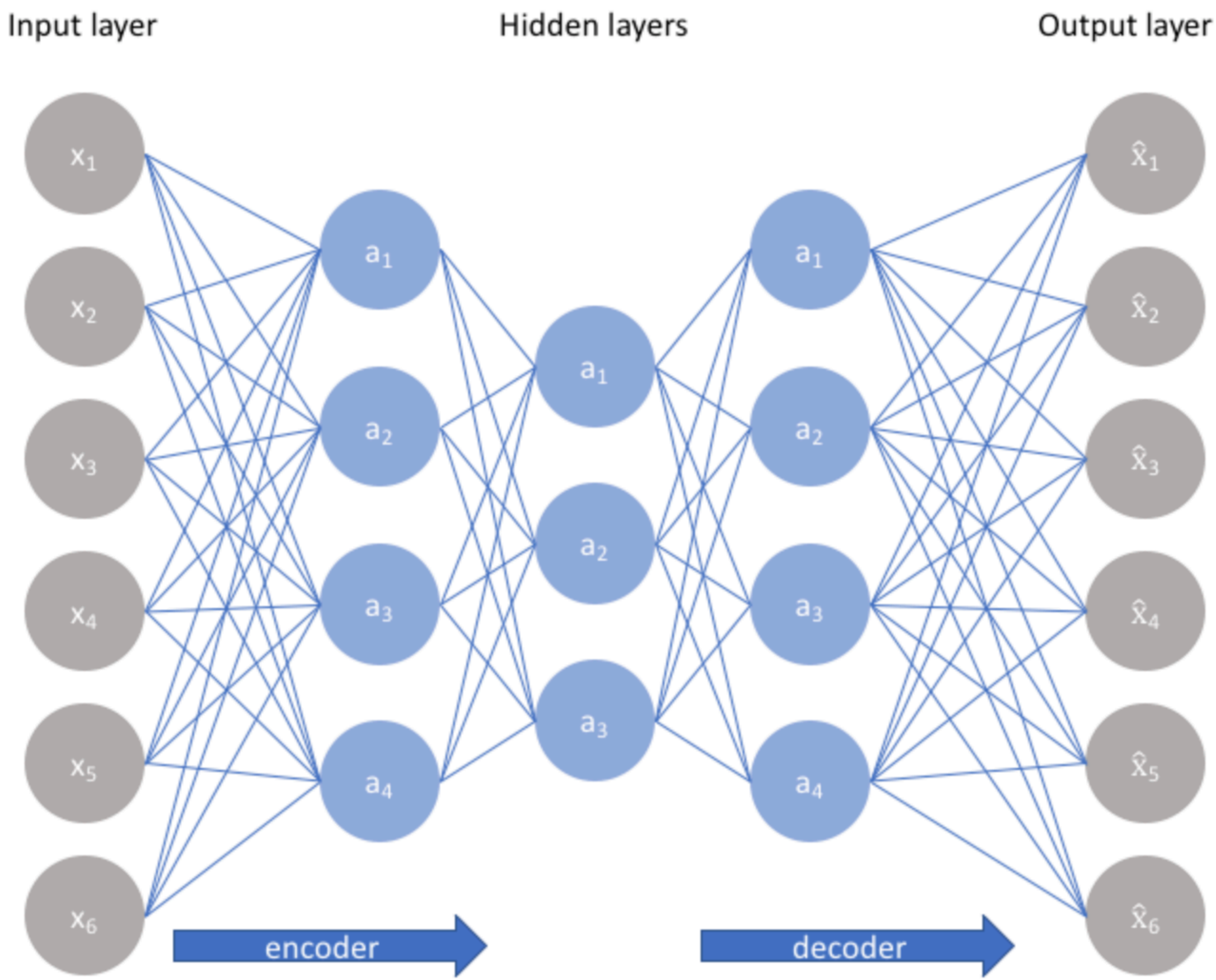
Auto-encoders: predict the **input** given the **input**.

Key idea: select features by **reducing then increasing** dimensionality.

Normal NN goes:

Auto-encoder goes:

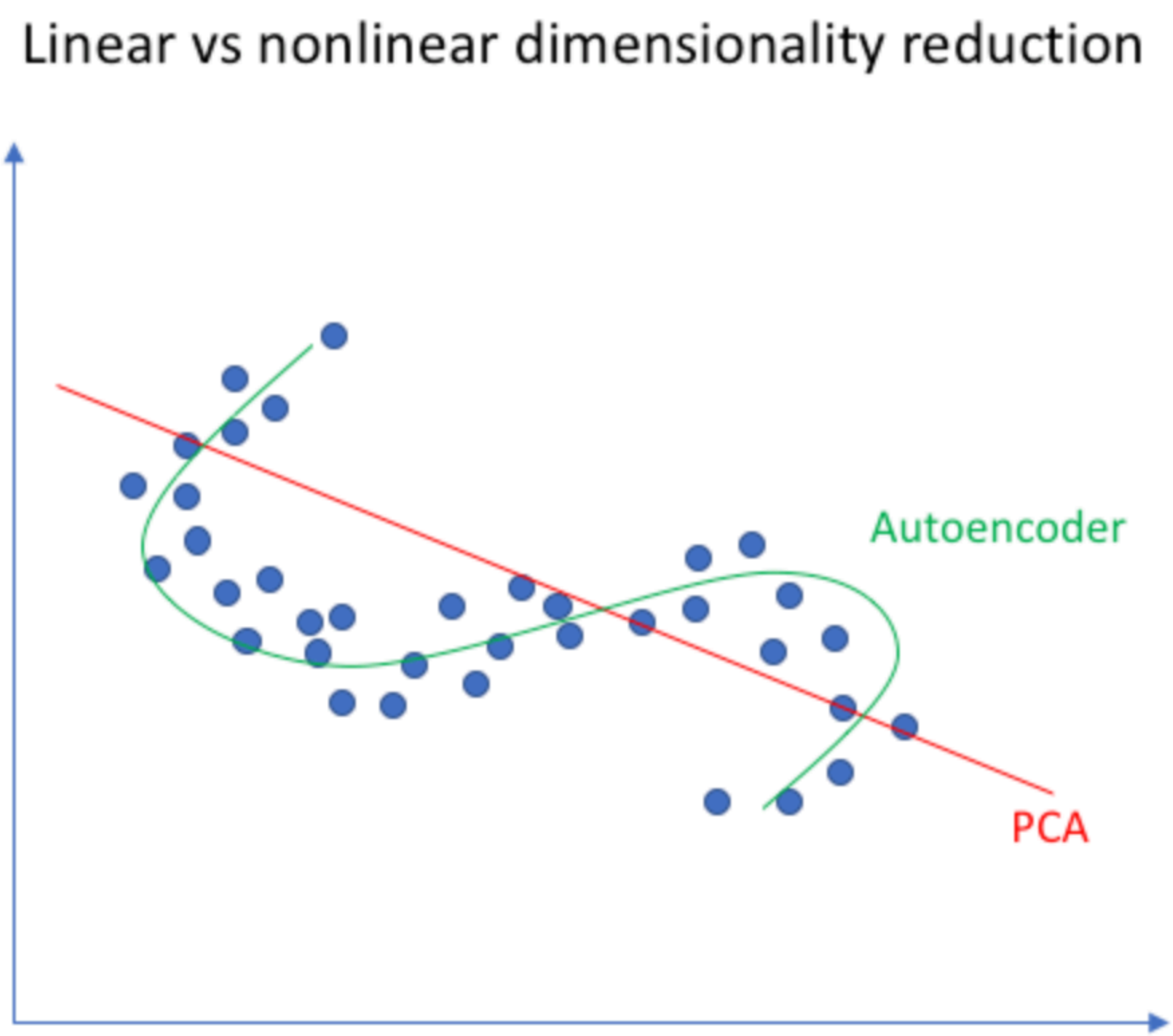
Auto-Encoder Architecture



www.jeremyjordan.me/autoencoders

Auto-Encoders as Dimensionality Reduction

Auto-encoders are a more powerful form of dimensionality reduction than traditional techniques like PCA, because they can learn nonlinear transformations.



www.jeremyjordan.me/autoencoders

Encoder

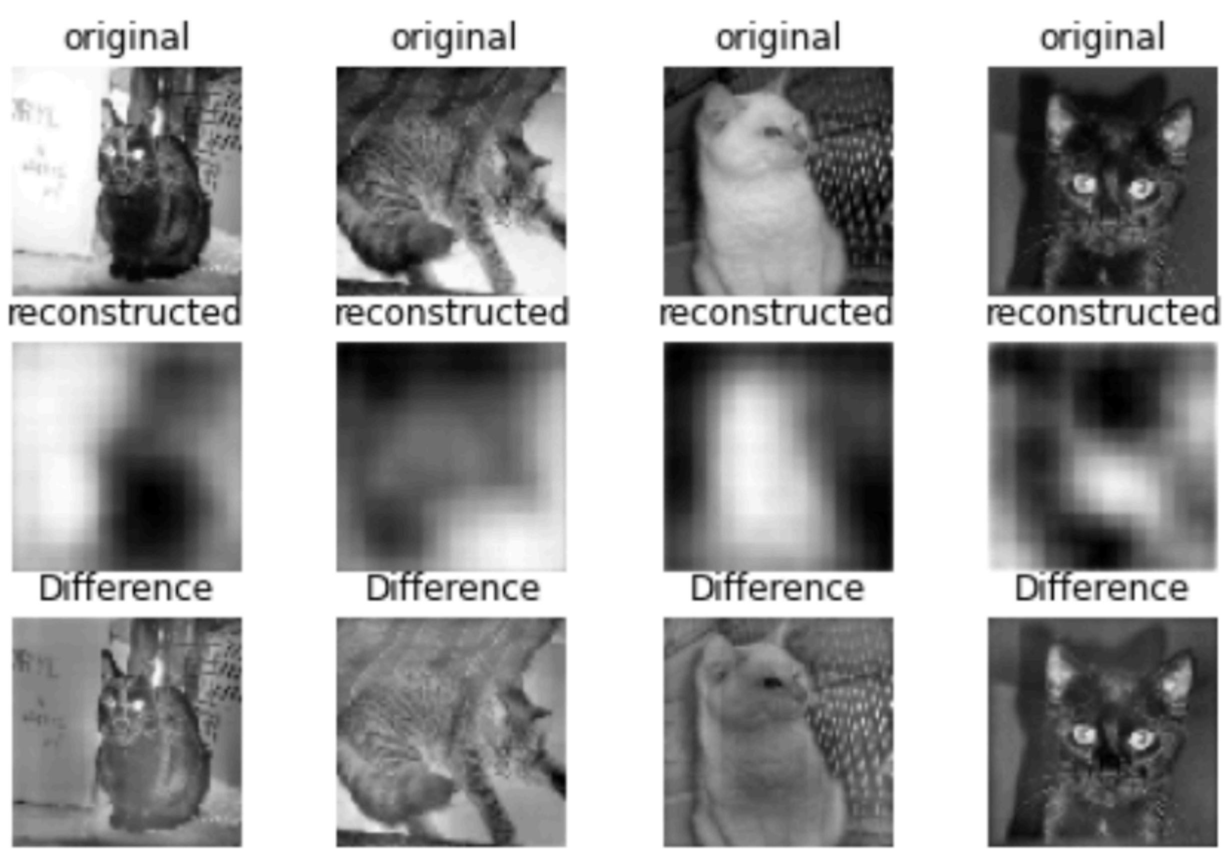
```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 32, 32, 32)        320

 dropout_18 (Dropout)        (None, 32, 32, 32)        0

 conv2d_10 (Conv2D)          (None, 16, 16, 64)        18496

 dropout_19 (Dropout)        (None, 16, 16, 64)        0

 conv2d_11 (Conv2D)          (None, 8, 8, 128)         73856

 dropout_20 (Dropout)        (None, 8, 8, 128)         0

 flatten_3 (Flatten)         (None, 8192)              0

 dense_6 (Dense)             (None, 128)               1048704

=================================================================
Total params: 1,141,376
Trainable params: 1,141,376
Non-trainable params: 0
_________________________________________________________________
```

Decoder

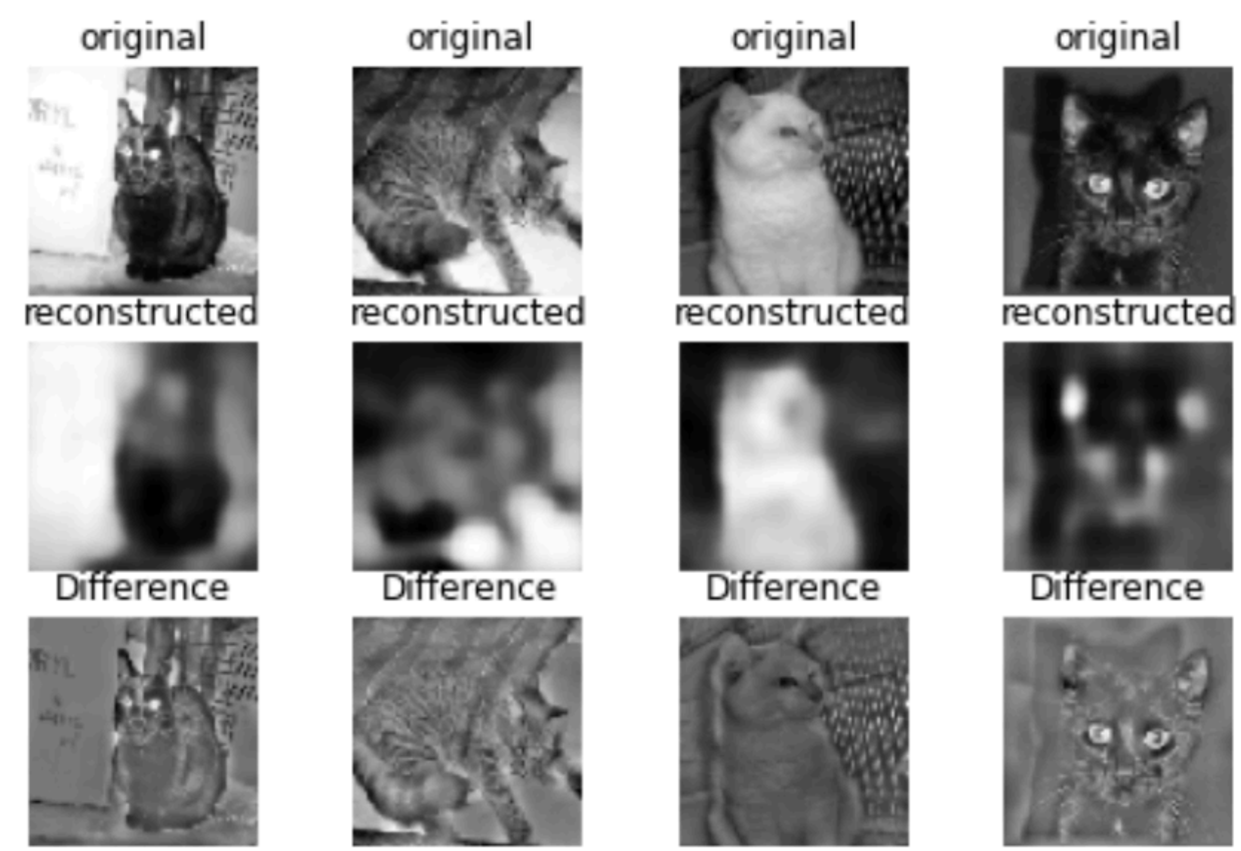
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_7 (Dense)             (None, 8192)              1056768

 reshape_3 (Reshape)         (None, 8, 8, 128)         0

 conv2d_transpose_12 (Conv2D  (None, 16, 16, 128)      147584
 Transpose)

 dropout_21 (Dropout)        (None, 16, 16, 128)       0

 conv2d_transpose_13 (Conv2D  (None, 32, 32, 64)       73792
 Transpose)

 dropout_22 (Dropout)        (None, 32, 32, 64)        0

 conv2d_transpose_14 (Conv2D  (None, 64, 64, 32)       18464
 Transpose)

 dropout_23 (Dropout)        (None, 64, 64, 32)        0

 conv2d_transpose_15 (Conv2D  (None, 64, 64, 1)        289
 Transpose)

=================================================================
Total params: 1,296,897
Trainable params: 1,296,897
Non-trainable params: 0
_________________________________________________________________
```

Input, Output, Difference

Epoch 1



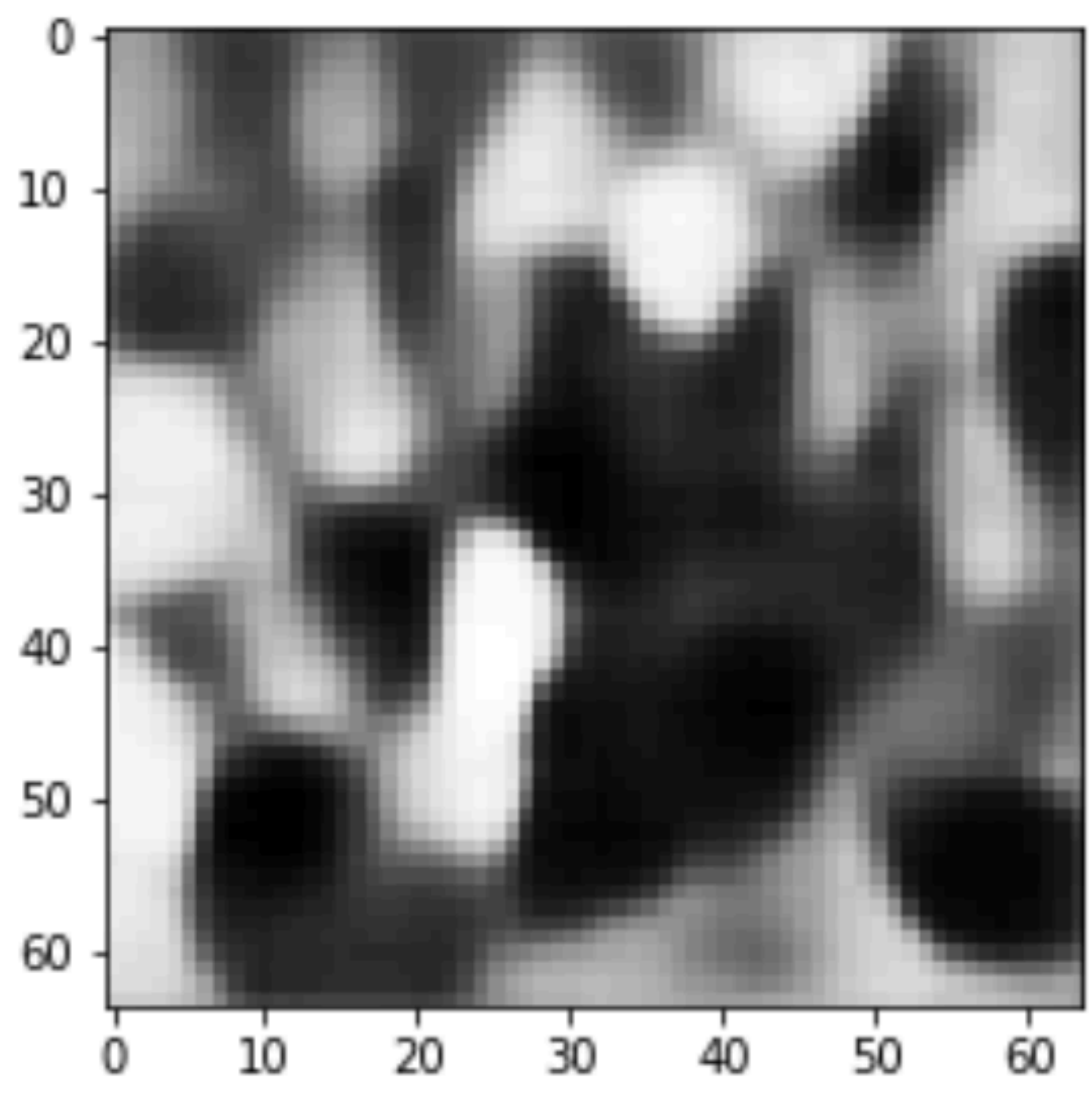
Epoch 10



Using Decoder to Generate

Input noise to the decoder to make it hallucinate a cat:

```
x = autoencoder.decoder(np.random.randn(1, 128)).numpy()
plt.imshow(x[0, :, :, 0], cmap='gray')
```



Stable Diffusion

Stable Diffusion

3 components:

1. VAE: an auto-encoder to map images to a latent space
2. U-Net: an architecture that learns to denoise images
3. CLIP: a text-encoder to allow multi-modal input

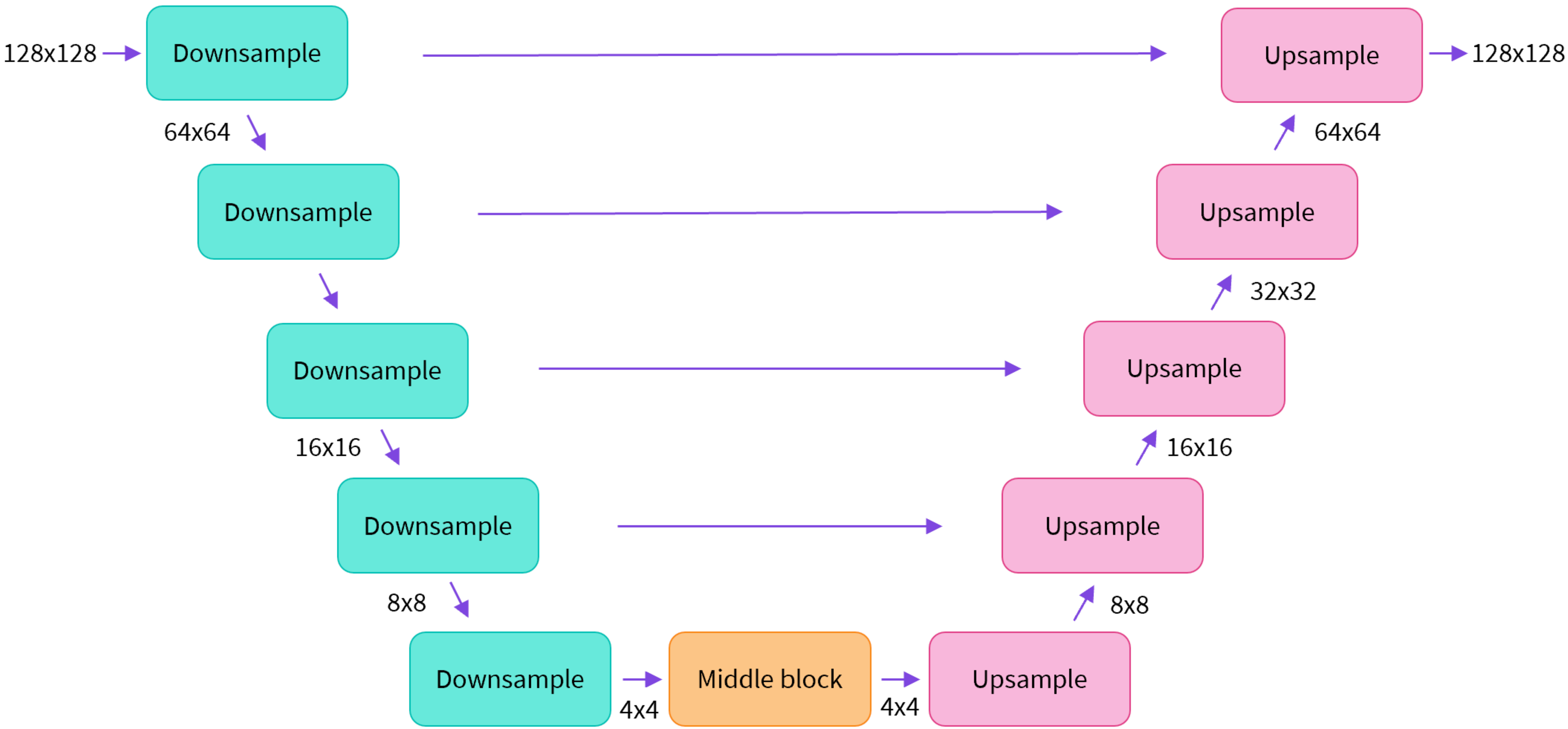
VAE: variational autoencoder

VAE is an encoder/decoder model.

The encoder maps an input image (pixels) to a lower-dimension latent space.

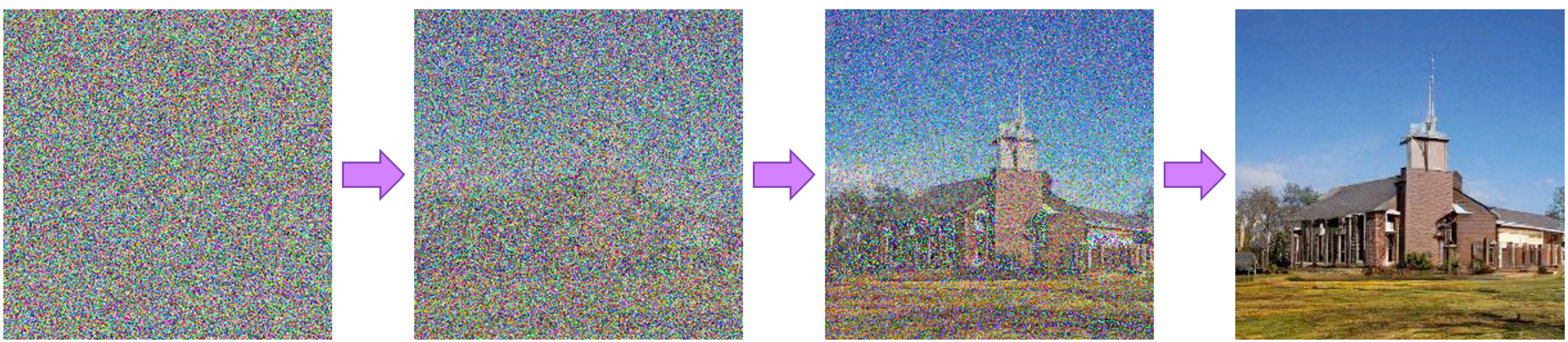
The decoder takes the output of the model and maps it back to an image in pixels.

U-Net model (auto-encoder)



https://colab.research.google.com/github/huggingface/notebooks/blob/main/diffusers/diffusers_intro.ipynb

Iteratively Denoising



CLIP: a text encoder for multi-modal input

Objective: given a batch of text and image inputs, predict the correct image-text pairings.

Learning Transferable Visual Models From Natural Language Supervision

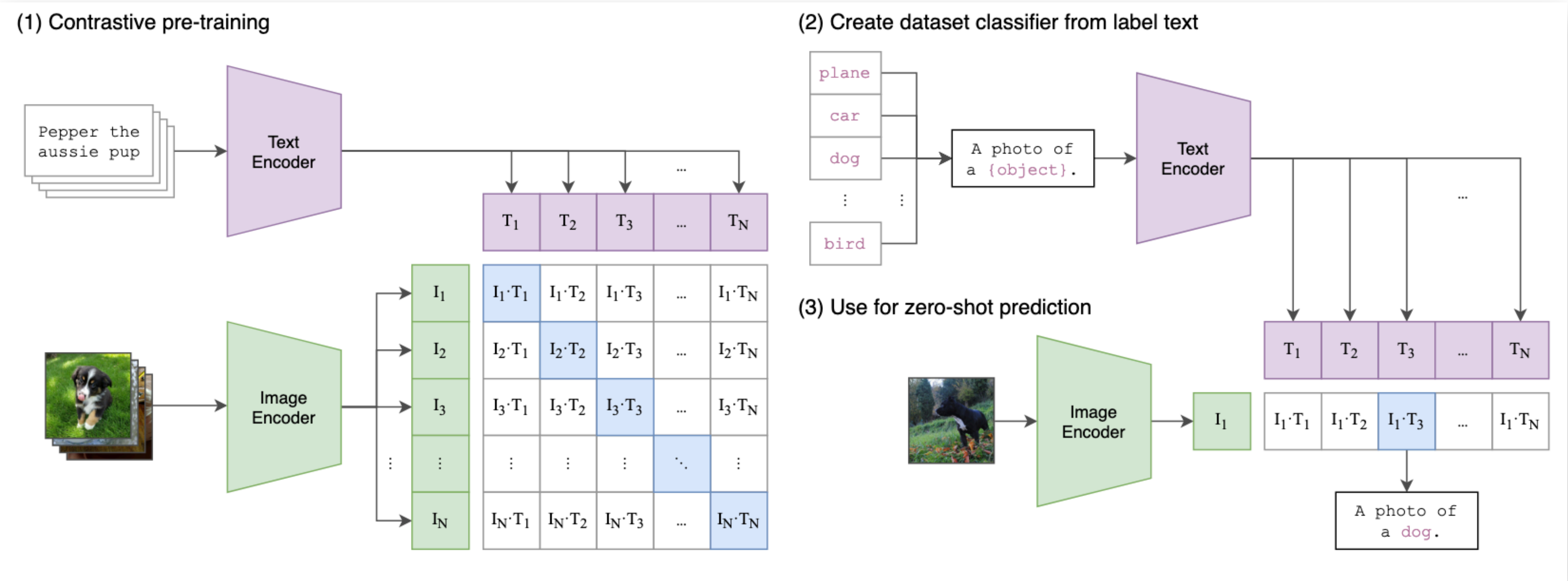
Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1] Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

Abstract

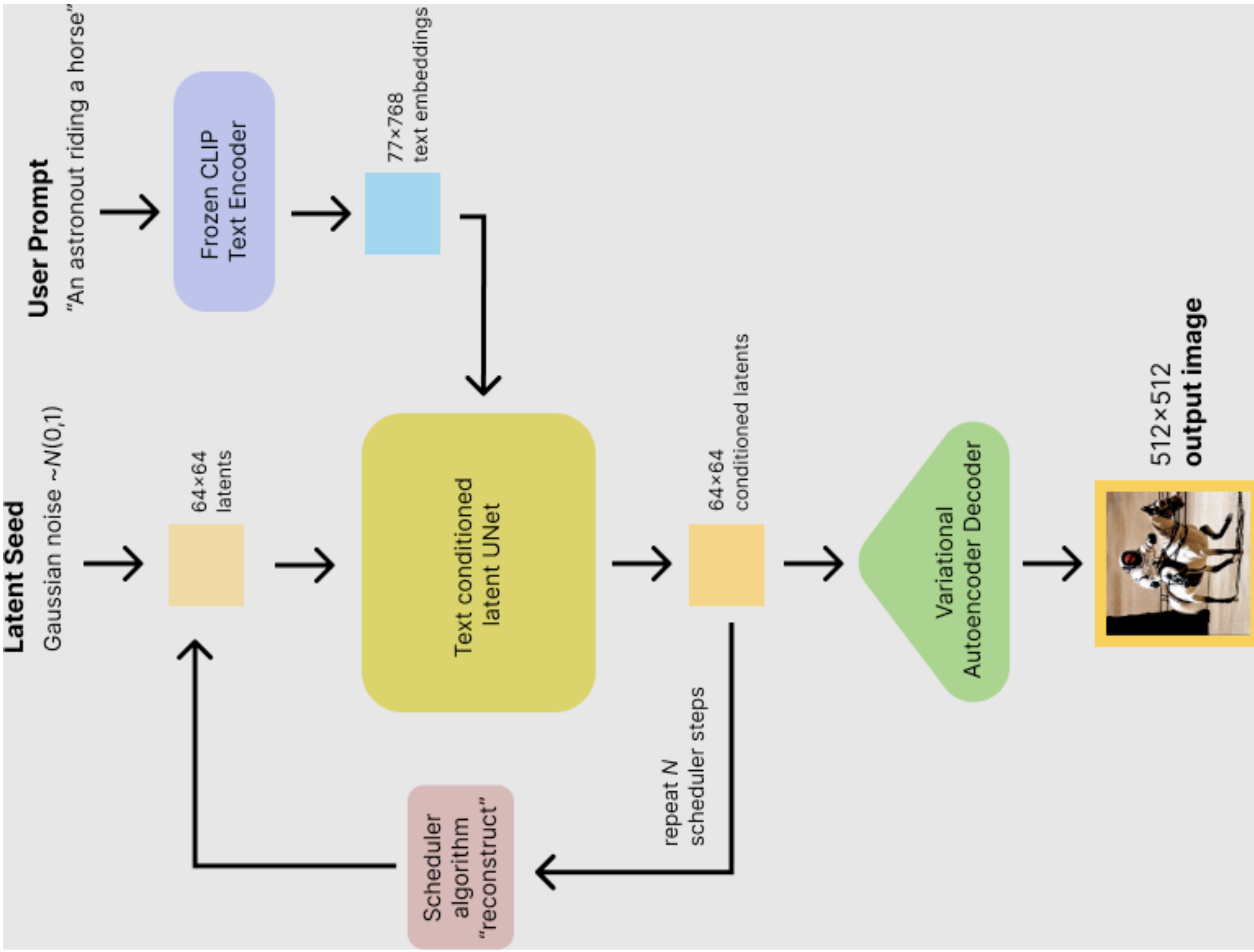
State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task

Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of "text-to-text" as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset

CLIP: a text encoder for multi-modal input



Stable Diffusion: putting the pieces together



repeat 50 times