Image Recognition with MNIST

Nimish Narang

ZENVA

Learning Goals

- How does image recognition work?
- What is MNIST?
- How do I build a machine learning model to recognize and classify images with Tensorflow?
- How do I build a machine learning model to recognize and classify images with Keras?

ZENVA

What do you Need Going in?

- Some Python and numpy knowledge
- A Python development environment (we will use Anaconda and Jupyter notebooks)
- An understanding of how machine learning works
- Some experience with Tensorflow
- Ideally, part 1 of this course (Intro to Machine Learning)

ZENVA

What Topics will we Cover?

- Intro to image recognition
- Intro to MNIST
- Build, train, and test an MNIST image recognition model
- Build, train, and test an MNIST image recognition model with Keras

ZENVA

Why Image Recognition?

- Image recognition is a fun and relatable topic
- There are many practical applications for recognizing and classifying images
- Once we understand how to build the model, it is easy to expand into other areas
- MNIST is an easy to use dataset

ZENVA

Intro to Image Recognition

ZENVA

What will we Cover?

- What is image recognition?
- What tools can help us to solve it?

ZENVA

What is Image Recognition?

- Image recognition is seeing an object or an image of an object and knowing what it is
- We essentially class everything that we see into certain categories based on attributes
- Even if we see something we have never seen before, we can usually place it in some category
- For example, if we see a brand new model of car for the first time, we can tell that it is a car by the wheels, hood, windshield, seats, etc.

ZENVA

How does this work for us?

- A lot of the time, image recognition for us happens subconsciously
- We don't necessarily acknowledge everything that is around us
- However, when we need to notice something, we can usually pick it out and define and describe it
- Knowing what something is is based entirely on previous experiences
- Some things we memorize, others we deduce based on shared characteristics that we see in things we do know

ZENVA

How does this work for us?

- Subconsciously we separate the items we see based on borders defined primarily by differences in colour
- Example: it is easy to see a green leaf on a brown tree but hard to see a black cat against a black wall
- We also don’t necessarily need to look at every part of an image to know what some part of it is
- Example: if we see only an eye and an ear of someone’s face, we know we are looking at a face

ZENVA

How does this work for Machines?

- Machines do not have infinite knowledge of what everything they see is
- They only have knowledge of the categories we have taught them
- For example, if you create facial recognition, it only classifies images into faces or not faces
- Even the most sophisticated image recognition models cannot recognize everything and have been trained only to look for certain objects

How does this work for Machines?

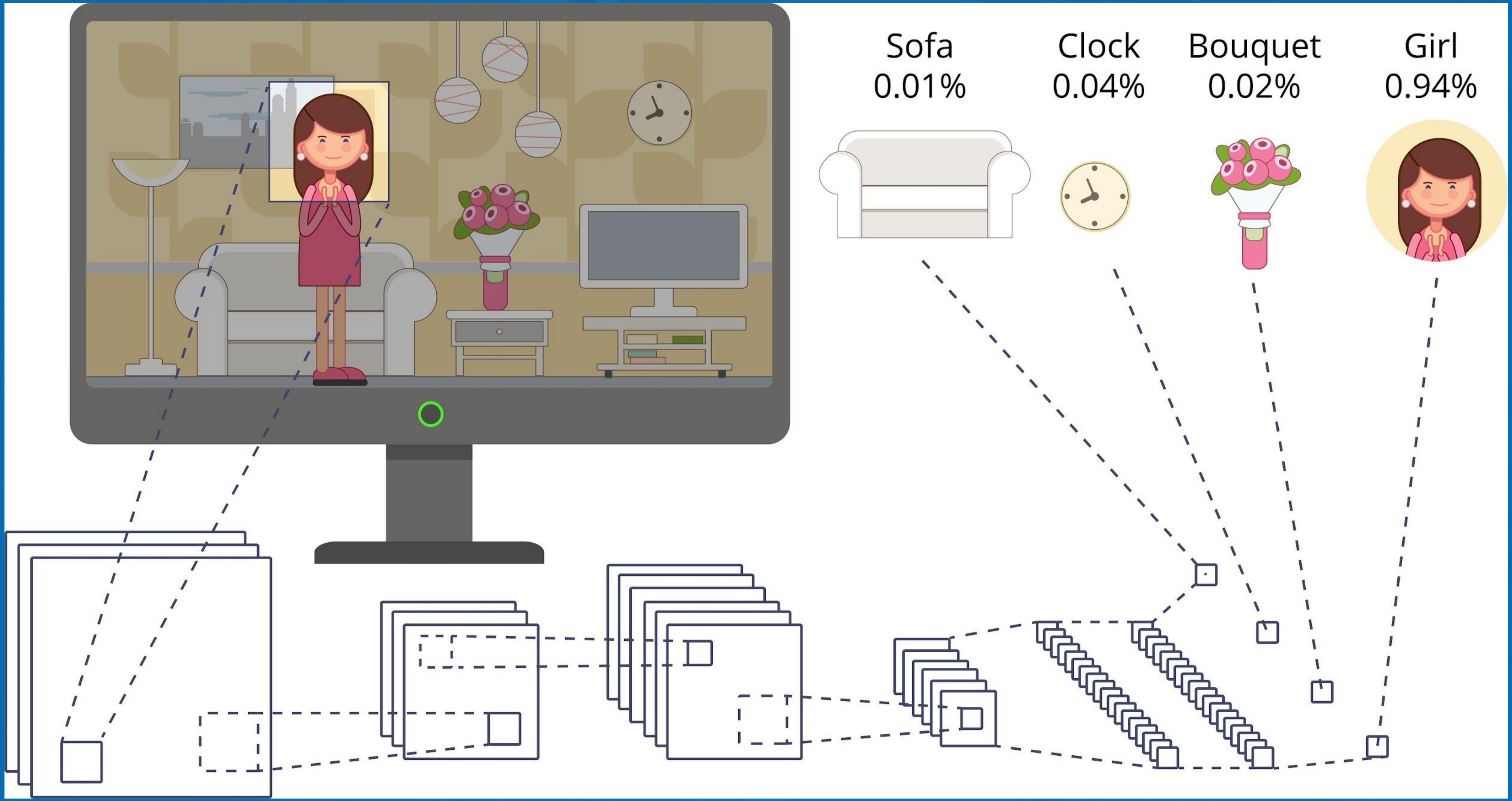
- To a machine, an image is simply an array of bytes
- Each pixel on an image contains information about red, green, and blue colour values
- If an image is just black or white, the value for each pixel is simply a darkness value, typically with 255 as white and 0 as black

ZENVA

How does this work for Machines?

- Machines don’t care about seeing an image as a whole
- To process an image, they simply look at the values for each of the bytes and look for patterns
- In this way, image recognition models look for groups of similar byte values across images to place an image in a specific category
- For example, high green and brown values in adjacent bytes may suggest an image contains a tree. If many images all have similar groupings of green and brown values, the model may think they all contain trees

How does this work for Machines?



ZENVA

Tools to Help with Image Recognition

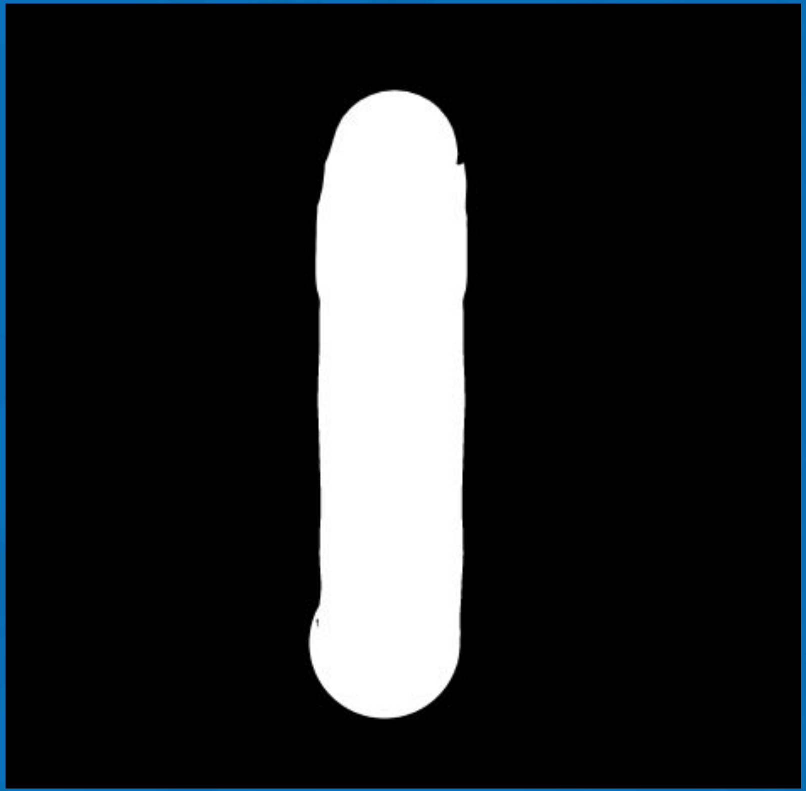
The Problem with Processing Images

- Processing an entire image at a time is a lot
- Most images fed into simple models are small (MNIST images are 28x28 pixels)
- Even this leaves 784 pixels to examine and it's difficult to recognize consistent patterns when comparing all 784 pixel values of one image to another
- Even if we are looking at two images of the same thing, slight position or size changes or slightly different shapes could lead to a mislabelling

ZENVA

Image as a Machine sees it

```
[[0,0,0,0,0],
 [0,0,255,0,0],
 [0,0,255,0,0],
 [0,0,255,0,0],
 [0,0,0,0,0]]
```



How Machines Solve this

- Machines solve this problem by first breaking down images into smaller parts and processing them instead
- It starts by applying a sort of a filter to different parts of the image a few pixels at a time to produce several smaller, distorted images
- Next, it makes the pieces more abstract by averaging together smaller squares of pixel values
- This ultimately turns this image into something that may not be recognizable by humans but the machines can make sense of it

Convolutional Neural Networks

- The process of breaking down images and applying the filters is done in a **convolutional layer** in a neural network
- Averaging the values to further distort images is done through a **max pooling layer**
- **Convolutional Neural Networks** get their name from the fact that they have one or more convolutional layers
- These are very popular in image recognition models, although RNNs have also performed well

ZENVA

Convolutions

- **Convolution:** an operation on two functions to produce a third function that explains how the shape of one is modified by the other
- **Image Convolution:** applying a **kernel** or **convolution mask** to blocks of pixels to apply an effect
- Often used to blur or sharpen images, or detect edges
- A **Kernel** is a matrix used as a mask to image pixel values
- Kernels are often pre-determined through Tensorflow objects

ZENVA

Convolution

Image:

[[0,0,0,0,0],
 [0,0,255,0,0],
 [0,0,255,0,0],
 [0,0,255,0,0],
 [0,0,0,0,0]]

Kernel:

[[0,-1,0],
 [-1,5,-1],
 [0,-1,0]]

First result:

[[, ,],
 [,-255,],
 [, ,]]

Max Pooling

- **Max Pooling**: replacing a block of pixels by one pixel with the highest value out of the block
- Makes an image more abstract
- Used to downsize an image and also prevent overfitting (drawing conclusions when there are none)
- Generally we specify the size of the matrix and the step size (number of pixels to skip over when applying the next max pool

Max Pooling

Original

[[5,10,15,10],
 [15,20,10,12],
 [13,16,9,8],
 [15,19,25,20]]

Final Result

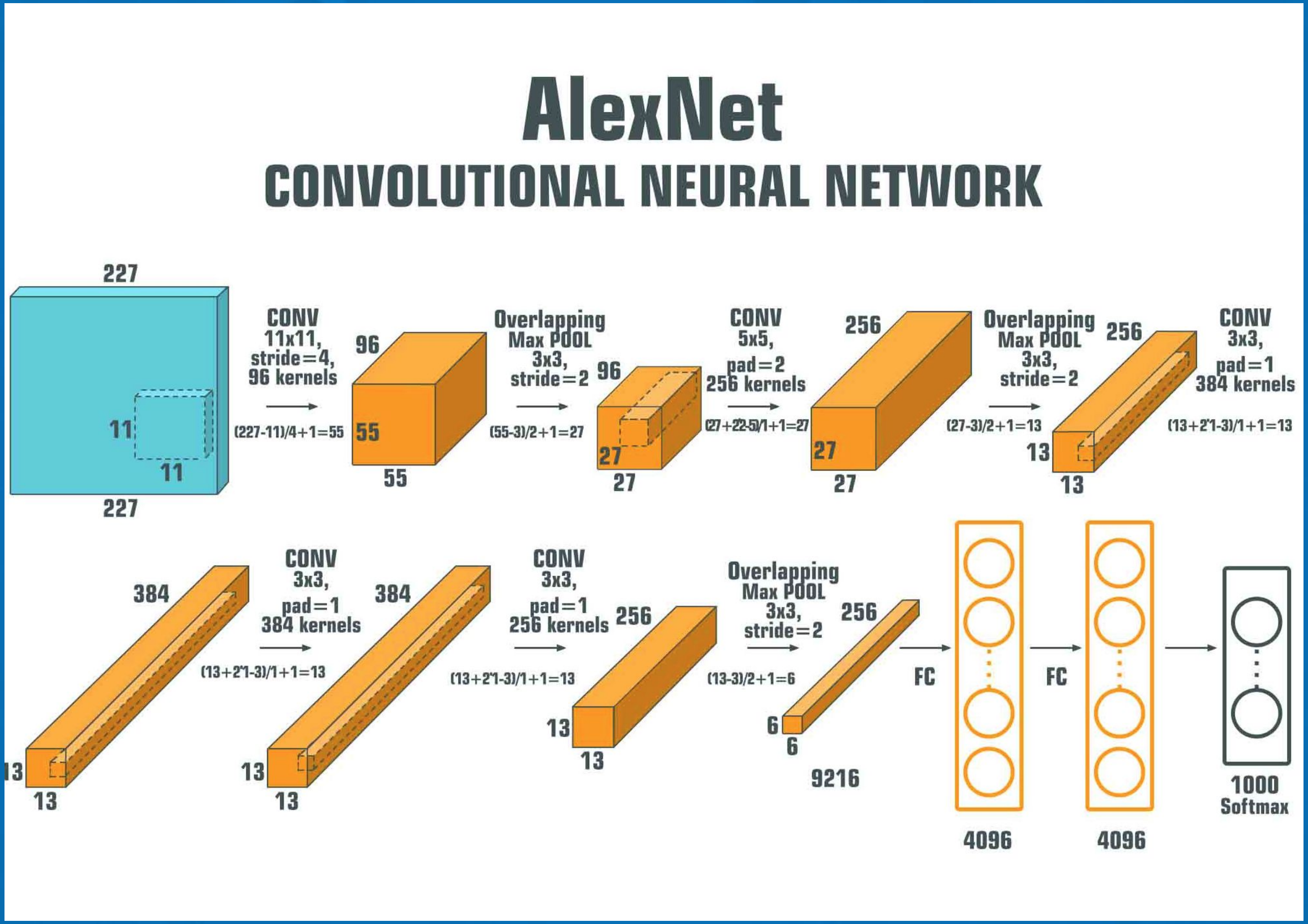
[[20,15],
 [19,25]]

ZENVA

Putting it Together

- Almost all convolutional neural networks will have a convolutional layer followed by a max pooling layer
- Larger networks will repeat this one or more times along with other layers to perform additional processing
- The result of the two layers is a set of smaller, more abstract images comprised of parts of the original image
- The purpose is to cut out unnecessary image noise and to focus on the stand-out features so that the model can focus on what is important

Putting it Together



ZENVA

MNIST

What is MNIST?

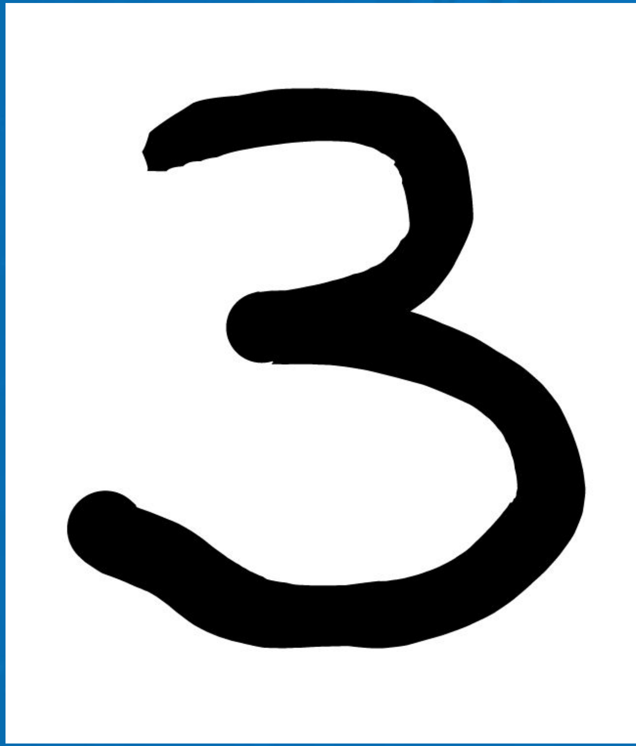
- MNIST = Modern National Institute of Standards of Technology
- We will use the dataset that MNIST is famous for
- The dataset contains 70,000 images of handwritten digits
- The even more modern EMNIST dataset was released in 2017 that contains 280,000 images

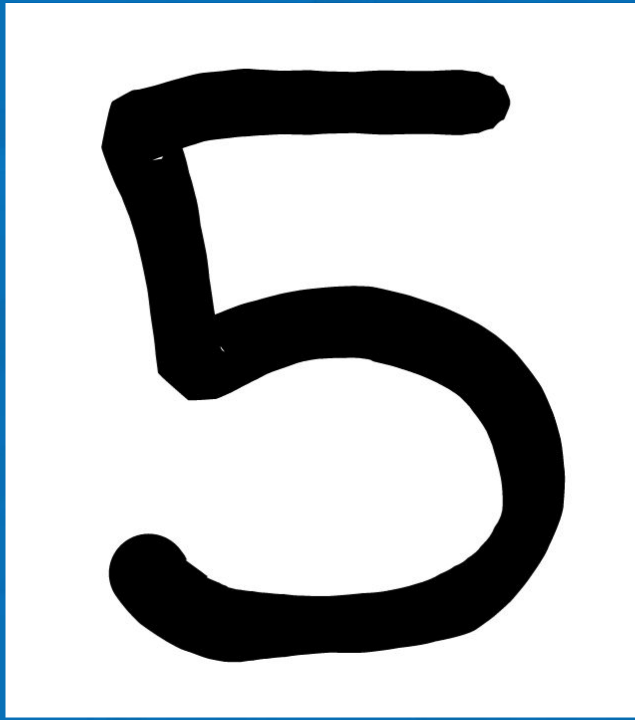
How are Images Formatted?

- 60,000 training images and 10,000 testing images
- Each image is 28x28 pixels (784 total)
- Each image is black and white
- Each image is labelled based on which image it represents
- Each label is in one-hot encoding form
- Instead of a string label, we have an array of 0s and 1s with the 1 in the position that represents the digit and 0s in the rest

ZENVA

How are the Images Formatted?





ZENVA

Why Bother with MNIST?

- Very highly esteemed data set
- Starting point for many image recognition models
- Great way to learn how to build an image recognition/classification model with a trusted and pre-formatted data set
- Ongoing competition to see who can get the best results (best so far is 0.21% error rate achieved by an ensemble of 5 CNNs produced by The Parallel Computing Center)

ZENVA

Building a CNN with Tensorflow Part 1

ZENVA

Part 1 - Obtaining Data

- Download MNIST dataset
- Examine images
- Examine labels

ZENVA

Building a CNN with Tensorflow Part 2

Part 2 - Layers

- Convolution layer
- Flatten layer
- Dense layer

Building a CNN with Tensorflow Part 3

ZENVA

Part 3 - Loss and Optimizer

- Add a loss function
- Add an optimizer function
- Add a way to measure loss and accuracy

Building a CNN with Tensorflow Part 4

ZENVA

Part 4 - Train and Test Step

- Add the function to run when training the model
- Add the function to run when testing the model

Building a CNN with Tensorflow Part 5

ZENVA

Part 5 - Formatting Data

- Format our inputs
- Format our outputs

Building a CNN with Tensorflow Part 6

Part 6 - Training

- Write the train loop
- Train and evaluate the model

ZENVA

Keras

What is Keras?

- **Keras** is a higher level machine learning library aimed at rapid construction and deployment of neural networks
- Provides a more abstract interface
- Takes a more more modular approach by grouping functionality into objects
- Can run with a Tensorflow, MCT, Theano, or PlaidML backend but we default and most popular is Tensorflow

Why use Keras?

- In short: Keras is easier to use than raw Tensorflow
- The modular approach makes it easy to build a model layer by layer
- Built in functions to train, test, and evaluate models
- Becoming more commonplace than raw Tensorflow
- Tensorflow 2.0 is closely integrated with Keras and calls upon Keras functionality in many models and datasets

How do we Obtain Keras?

- Keras should already come with Tensorflow
- Can be found under the module tf.keras
- Many parts of the Tensorflow ecosystem are now tf.keras.something

Building a CNN with Keras

ZENVA

Steps

- Obtain the data
- Add the model
- Add the layers
- Train the model
- Test the model

Image Recognition with MNIST

Nimish Narang

ZENVA

Learning Goals

- How does image recognition work?
- What is MNIST?
- How do I build a machine learning model to recognize and classify images with Tensorflow?
- How do I build a machine learning model to recognize and classify images with just Keras?

ZENVA

What Topics did we Cover?

- Intro to image recognition
- Intro to MNIST
- Build, train, and test an MNIST image recognition model
- Build, train, and test an MNIST image recognition model with Keras

Where to go from here

- Improve upon our model by modifying structure
- Try other image recognition data sets like CIFAR
- Try image recognition with RNNs or other types of neural networks
- Explore other types of neural networks and try solving different problems with them
- Keep exploring the Tensorflow and Keras ecosystem!

ZENVA