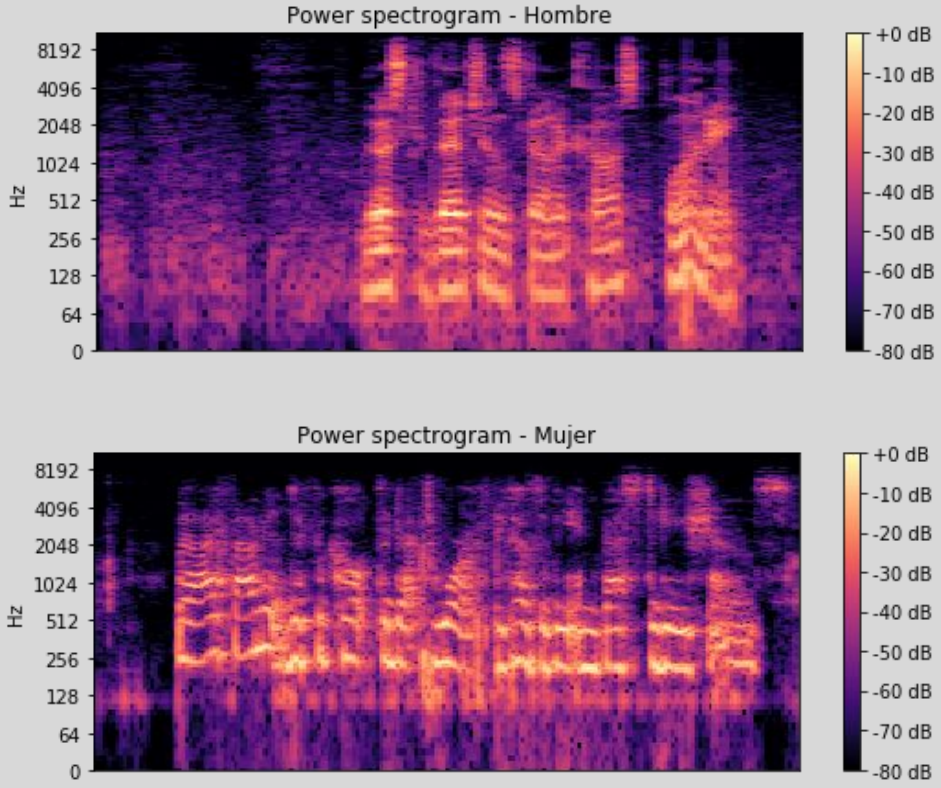


DigitalHouse > Coding School

INTELIGENCIA ARTIFICIAL

Clase final

OBJETIVOS DE LA CLASE

1. Audio
2. Attention
3. Lo que se viene
4. IA en el mundo real

Audio

Vamos a ver un caso de aplicación sobre un dataset de grabaciones de personas creado por mozilla.

https://www.kaggle.com/mozillaorg/common-voice

El dataset tiene un subconjunto de datos validados, y dentro de este, una división entre test, train y development. Sobre este último conjunto de datos trabajaremos, aunque, naturalmente, en un trabajo para producción, deberíamos utilizar el set de train.
Si bien no todos los datos están etiquetados por género, del conjunto de development hay unos 1500 que sí lo están.

Preprocesamiento

Pipeline general

- Preprocesamiento:
  - Silencios, normalizar, comprimir, filtrar ruido...
  - Transformadas:
    - Fourier, Cosine
    - Mel, Bark
    - Wavelet
  - Chunking
- Separación train, test (ojo!):
  - Eq
  - Canción
- Posibles representaciones

Pasos:

1. Cargar audio
2. eliminar silencios
3. Calcular el MFCC
4. recortar el audio hasta el maxlength
5. Normalizar el audio
6. Realizar el Padding
7. Rotar
8. Agregar una dimensión al tensor

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

1. Cargar audio:
    - utilizamos la librería librosa y especificamos el sample rate deseado.
    - Es importante que todos los audios tengan el mismo sample rate

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

2. eliminar silencios:

```
def remove_silence(self,y,audio_path,label, sr):

        y, sr = librosa.load(audio_path)

        tmp_path = "/tmp/" + label + 'tmp.wav'
        tmp_path2 = "/tmp/" + label + 'tmp2.wav'

        librosa.output.write_wav(tmp_path,y,sr)
        result = check_output("sox " + tmp_path + " " + tmp_path2 + " " +
        "silence 1 0.1 -59.0d -1 2.0 -59.0d", shell=True)

        y, sr = librosa.load(tmp_path2, sr = sr)
        return y
```

En muchos problemas de clasificación de audio el silencio no aporta información relevante, pero *ocupa espacio*.
Por eso, es una buena práctica en estos casos eliminar el silencio

Pasos:

3. Calcular el MFCC:
    - Recuerden que el parámetro **n_mfcc** define la cantidad de bandas mel. Por lo general se utilizan entre 20 y 40, y depende de lo que requiera la arquitectura de la red

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

4. recortar el audio hasta el maxlength:
    - El shape del MFCC será de **(n_mfcc *x* T),** donde T es proporcional al largo en tiempo del audio
    - Todos los tensores (observaciones) del input deben tener igual formato.**(n_mfcc *x* T*)**
    - Aquellos que se exceden del maxlength, T*, los cortamos

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

5. Normalizar el audio:
    - Reescalamos los datos de MFCC entre -1 y 1
    - Al normalizar el audio, evitamos las diferencias de nivel entre audios, que no son diferencias propias de aquello que se quiere clasificar
        - Por ejemplo, si cambia el tipo de micrófono, puede haber un cambio de nivel, pero la relación de los elementos de la matriz mantenerse

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

6. Realizar el Padding:
    - El shape del MFCC será de **(n_mfcc *x* N),** donde N es proporcional al largo en tiempo del audio
    - Todos los tensores (observaciones) del input deben tener igual formato.**(n_mfcc *x* N*)**
    - Aquellos que no llegan a esa cantidad de columnas, les agregamos 0 al final, hasta llegar a N*

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

7. Rotar:
    - La rotación de los datos a un shape **(T* *x* n_mfcc)** es necesaria para entrenar una red de tipo LSTM, ya que la dimensión temporal son las filas.
    - En una CNN no es necesario rotar los datos, pero no empeora la performance

```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Pasos:

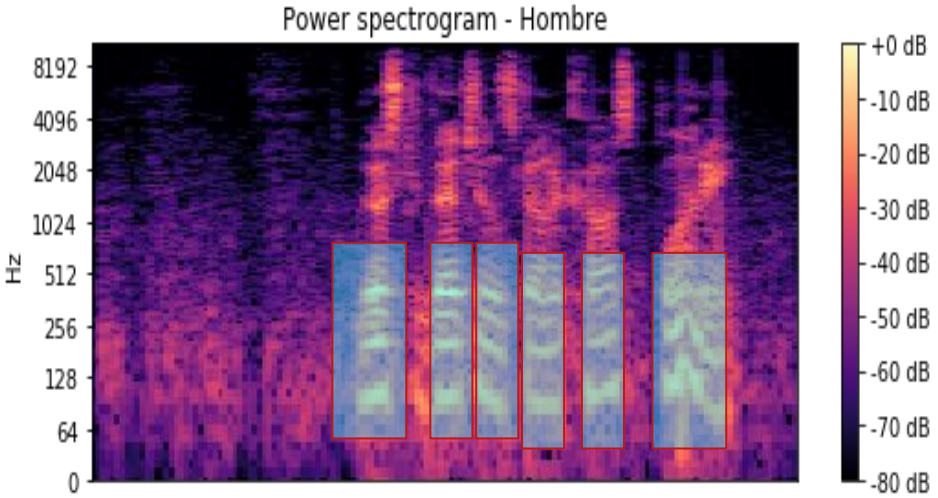
8. Agregar una dimensión al tensor:
    - Agregar una dimensión al tensor, para que el shape sea **(T* *x* n_mfcc *x* 1)**, esto es necesario para la CNN, ya que es como una imagen de un sólo canal.
    - En la LSTM no es necesario agregar esta dimensión, pero no afecta la performance

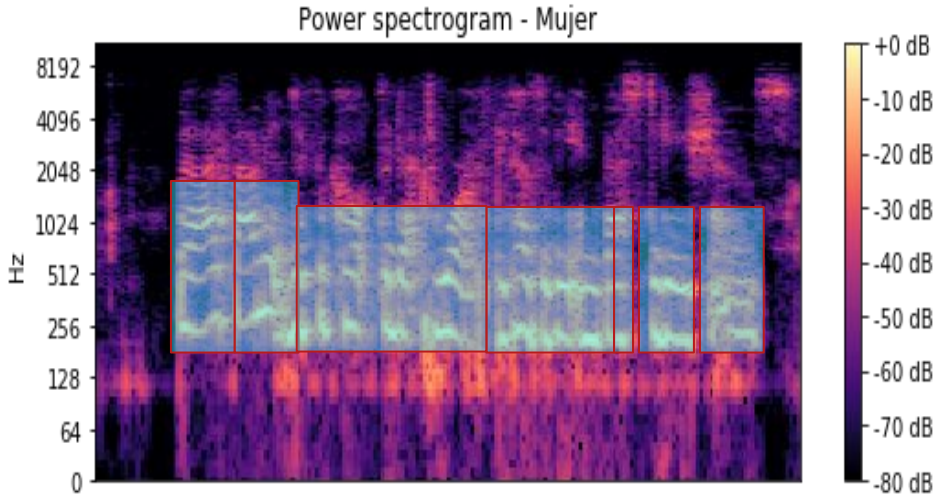
```
y, sr = librosa.load(audio_path, sr = sr)
y = self.remove_silence(y, audio_path, label, sr)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
mfcc = mfcc[:,:maxlength]
mfcc_pad = librosa.util.pad_center(librosa.util.normalize(mfcc), maxlength)
mfcc_rotate = np.rot90(mfcc_pad)
mfcc_rotate = mfcc_rotate.reshape((maxlength, mfcc_rotate.shape[1],1))
```

Clasificación con DNN

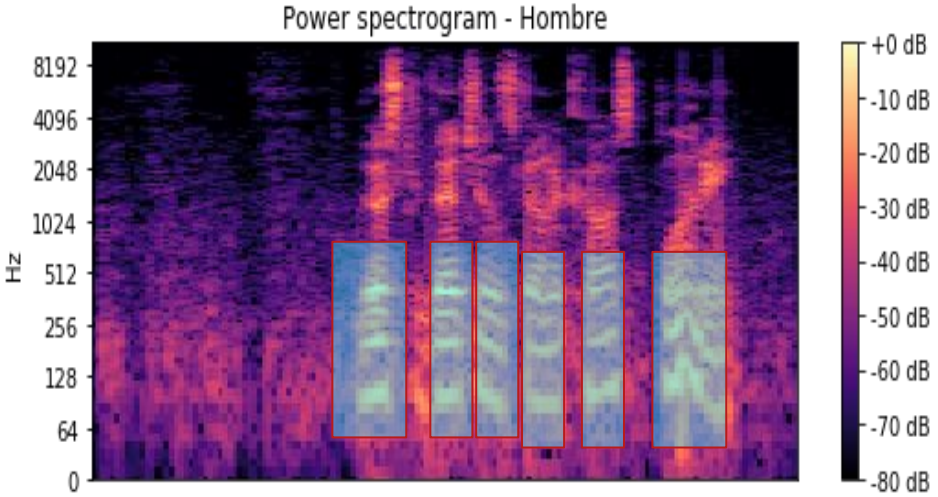
Ejemplo: Clasificación de género a partir de una grabación de una persona hablando.

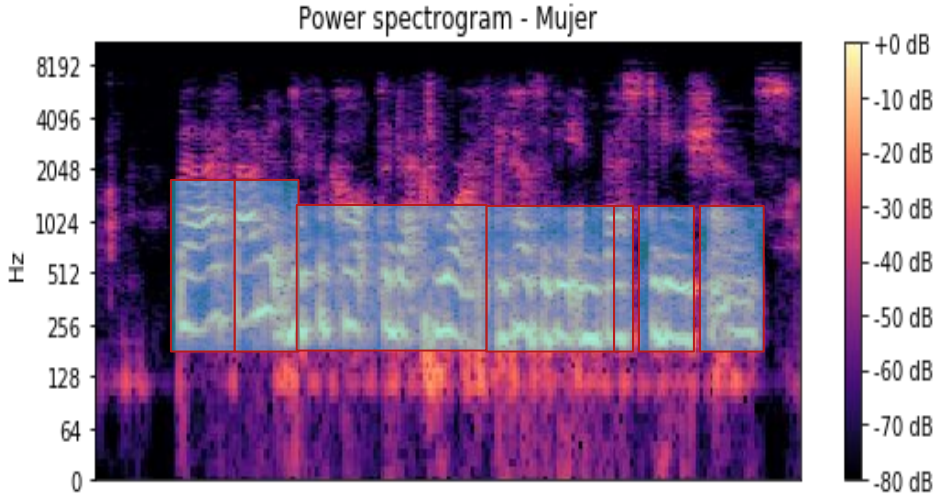
Es fundamental reconocer **¿qué tipo de patrones están involucrados en este problema?**



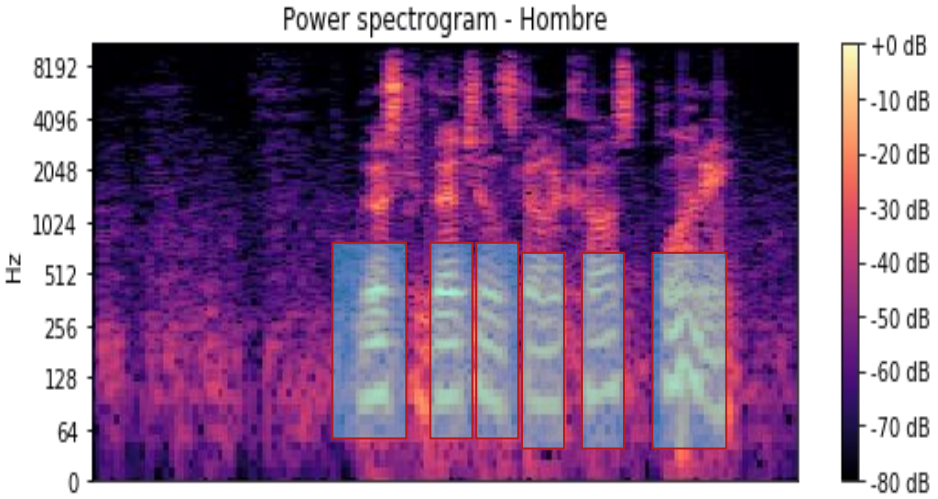


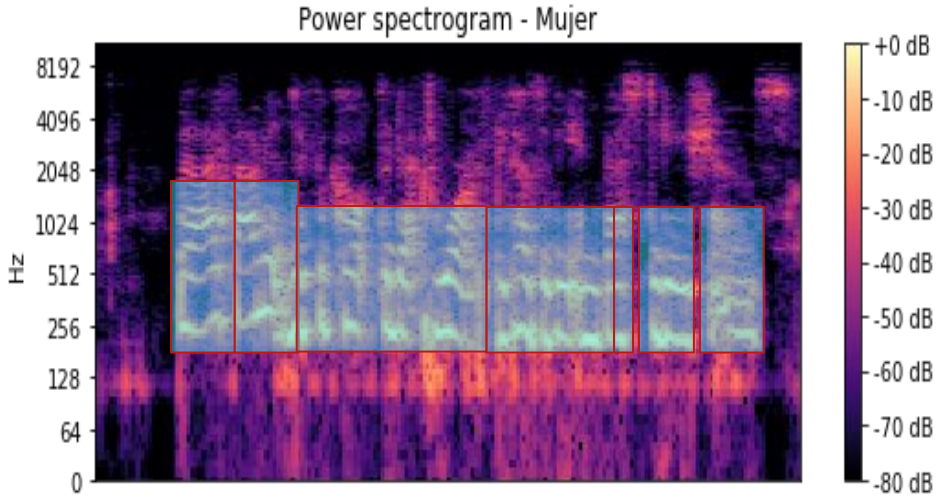
- Reconocer el tipo de patrón a detectar nos va a permitir elegir la mejor arquitectura de la red neuronal:
  - Si involucra detectar un movimiento: una secuencia de frecuencias de onda en el tiempo → necesitamos una red secuencial (ej. LSTM)
  - Si el movimiento en el tiempo no es tan importante, sino que el problema se asemeja a reconocer un objeto en una imagen → podemos utilizar una red convolucional





- En el caso de detección de género, se sabe que el registro vocal de las mujeres es más alto. Por lo tanto, podríamos reconocer que va a haber mayor potencia en frecuencias de onda mayores
    - Una CNN debería ser una buena aproximación del problema





Proponemos una arquitectura de CNN:

1 2 3 4 5

Convolución 2D:
- ➔ filters: 32
- ➔ kernel: (4,4)
- ➔ activación: relu

MaxPooling2D:
- ➔ pool_size: (3,3)
- ➔ stride: (2,3)

Dropout

Convolución 2D:
- ➔ filters: 64
- ➔ kernel: (3,3)
- ➔ activación: relu

MaxPooling2D:
- ➔ pool_size: (3,3)
- ➔ stride: (1,3)

Dropout

Convolución 2D:
- ➔ filters: 128
- ➔ kernel: (3,3)
- ➔ activación: relu

MaxPooling2D:
- ➔ pool_size: (3,3)
- ➔ stride: (2,3)

Dropout

Dense:
- ➔ units: 128
- ➔ activación: softmax

Dropout x 2

Dense::
- ➔ units: 2
- ➔ activación: softmax

CNN shapes

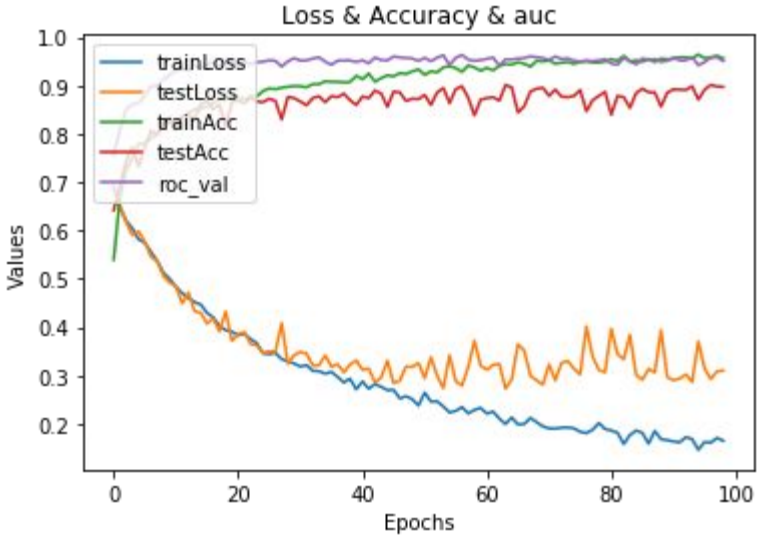
Input original: **40 x 300**

Total params: 912,482

1	2	3	4	5
Convolución 2D: kernel: (4,4): ➔ -3,-3 **MaxPooling2D:** stride: (2,3) ➔ /2,/3 **Output Shape:** **18 x 99 x 32** **Param #:** **544**	**Convolución 2D**: kernel: (3,3) ➔ -2,-2 **MaxPooling2D:** stride: (1,3) ➔ -2,/3 **Output Shape:** **14 x 32 x 64** **Param #:** **18496**	**Convolución 2D**: kernel: (3,3) ➔ -2,-2 **MaxPooling2D:** stride: (2,3) ➔ (-2,/2),/3 **Output Shape:** **5 x 10 x 128** **Param #:** **73856**	**FLATTEN** **Output Shape:** **128** **Param #:** **819328**	**no cambia** **Output Shape:** **2** **Param #:** **258**

```
def convModel(rows,cols, num_classes, dropout):
    model = Sequential()

    model.add(layers.Conv2D(32, (4, 4), activation='relu', input_shape=(rows,cols,1)))
    model.add(layers.MaxPooling2D((3, 3), strides = (2,3)))
    model.add(layers.Dropout(dropout))

    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides = (1,3)))
    model.add(layers.Dropout(dropout))

    model.add(layers.Conv2D(128, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((3, 3), strides = (2,3)))
    model.add(layers.Dropout(dropout))

    model.add(layers.Flatten())
    model.add(layers.Dense(128, activation='softmax'))
    model.add(layers.Dropout(2*dropout))
    model.add(layers.Dense(num_classes, activation='softmax'))
    model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Resultados:



Matriz de confusión:

334	52
35	351

Área debajo de la curva ROC = 0.95
Accuracy en Test = 0.90

Podemos comparar este ejercicio con una alternativa de Machine Learning:

- Los algoritmos de ML no pueden recibir como input un tensor
- Tengo que reducir una dimensión
- Para ello, como nuestros datos originalmente tienen una dimensión **(T* *x* n_mfcc)**, podemos colapsar una de las dimensiones, utilizando una medida de resumen:
    - Por ejemplo, la dimensión temporal se puede colapsar en la media.

Para calcular las medidas de resumen en XGBoost ¿deberíamos hacer zero-padding? ¿Por qué

Resultados XGBOOST

Accuracy en test: 0.86
AUROC: 0.8

	precision	recall	f1-score	support
0	0.90	0.91	0.91	386
1	0.71	0.68	0.70	119
avg / total	0.86	0.86	0.86	505

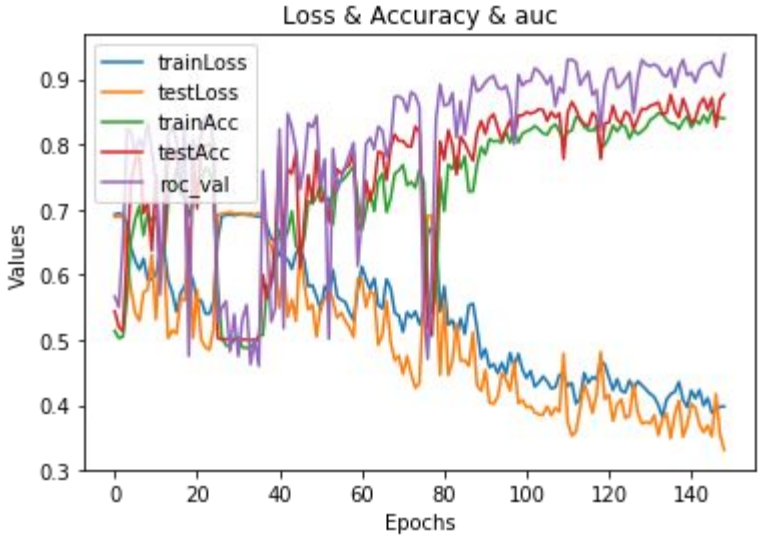
Matriz de confusión:

353	33
38	81

¿Qué pasa si utilizamos una arquitectura del tipo LSTM?

```
def lstmModel(timesteps, data_dim, num_classes, input_neurons, dropout,recurrent_dropout):
    model = Sequential()
    model.add(LSTM(input_neurons, return_sequences=True,
                   input_shape=(timesteps, data_dim),
                   dropout = dropout, recurrent_dropout = recurrent_dropout))

    model.add(LSTM(input_neurons, return_sequences=True,
                   dropout=dropout, recurrent_dropout=recurrent_dropout))

    model.add(LSTM(input_neurons, return_sequences=True,
                   dropout=dropout, recurrent_dropout=recurrent_dropout))

    model.add(LSTM(input_neurons, dropout=dropout, recurrent_dropout=dropout))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model
```

Resultados:



Matriz de confusión:

345	41
52	334

Área debajo de la curva ROC = 0.94
Accuracy en Test = 0.88

Resultados:

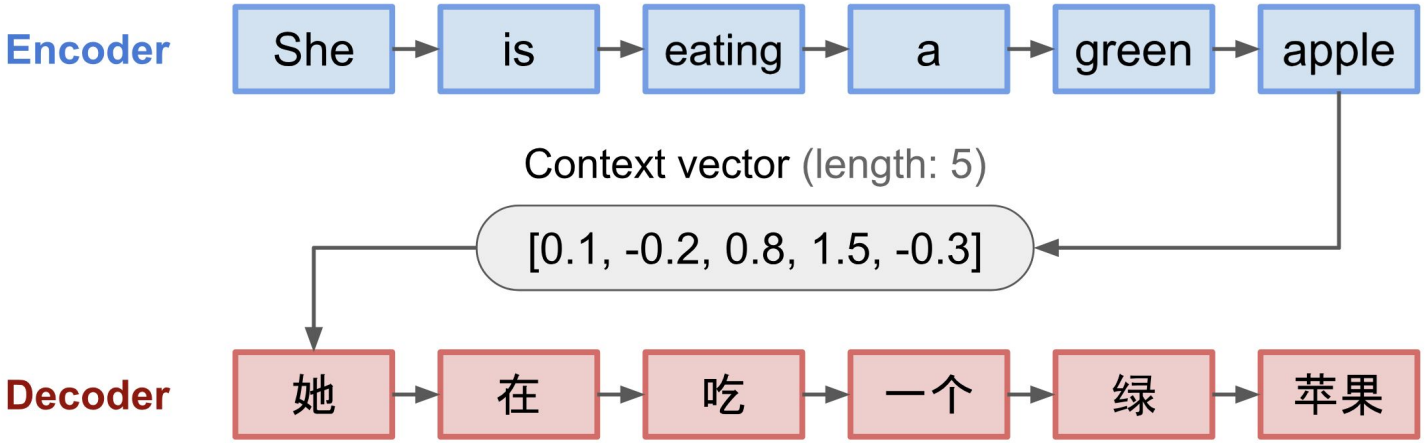
Modelo/ Métrica	Accuracy en Test	Área ROC
XGBOOST	0.86	0.8
LSTM	0.88	0.94
CNN	**0.90**	**0.95**

Práctica Independiente

Attention

seq2seq

- Modelos encoder-decoder con RNN



Attention

- Soft attention / Hard attention
- Global / Local attention

by *ent423* , *ent261* correspondent updated 9:49 pm et , thu march 19 , 2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* , *ent85* , near *ent312* , a *ent119* official told *ent261* on wednesday . he was identified thursday as special warfare operator 3rd class *ent23* , 29 , of *ent187* , *ent265* . `` *ent23* distinguished himself consistently throughout his career . he was the epitome of the quiet professional in all facets of his life , and he leaves an inspiring legacy of natural tenacity and focused

. . .

ent119 identifies deceased sailor as **X** , who leaves behind a wife

by *ent270* , *ent223* updated 9:35 am et ,

(*ent223*) *ent63* went familial for fall at it

ent231 on sunday , dedicating its collecti

with nary a pair of `` mom jeans '' in sight

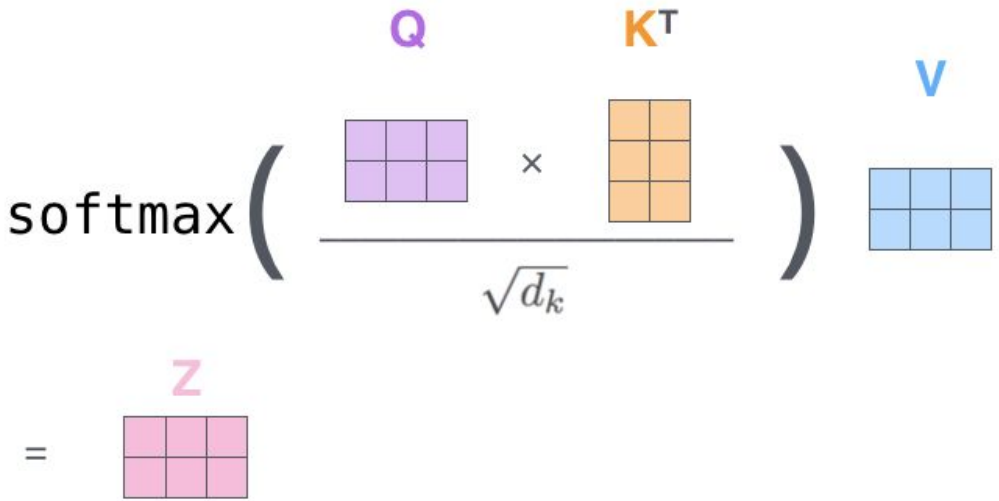
who are behind the *ent196* brand , sent models down the runway in decidedly feminine dresses and skirts adorned with roses , lace and even embroidered doodles by the designers ' own nieces and nephews . many of the looks featured saccharine needlework phrases like `` i love you ,

. . .

X dedicated their fall fashion show to moms

Attention

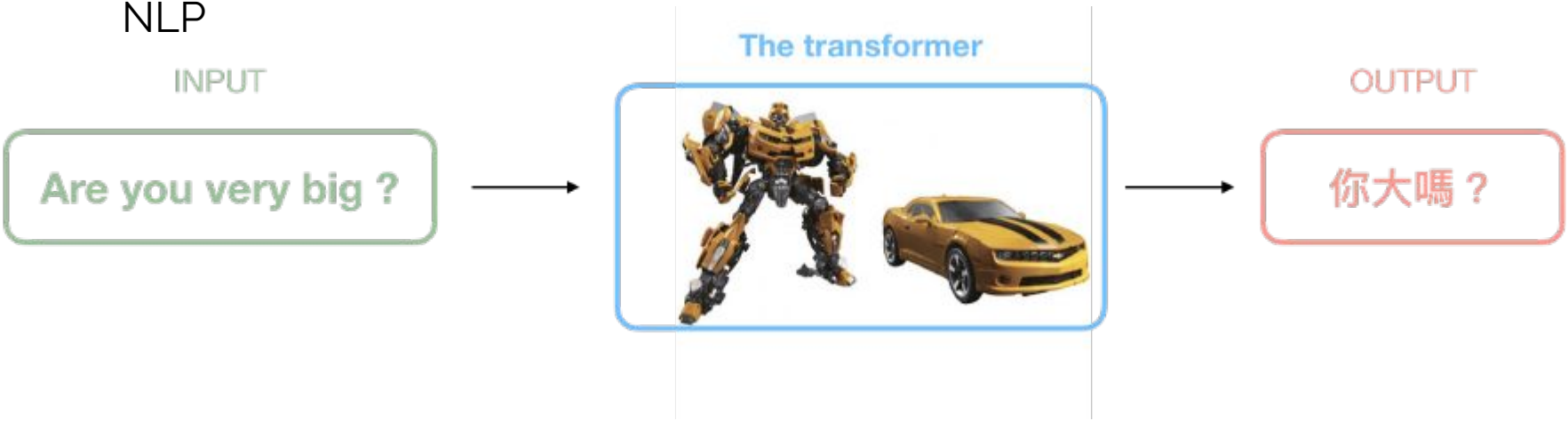
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



The self-attention calculation in matrix form

The Transformer

- Arquitectura propuesta en dic de 2017 en "Attention is all you need"
- Superó el estado del arte en múltiples tareas, especialmente en NLP



https://magenta.tensorflow.org/music-transformer

Bert

- Deeply bidirectional
    - Preentrena para predecir palabras maskeadas
- Transformers:
    - Encoders-decoders con attention

SQuAD1.1 Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

Lo que se viene

Neural Ordinary Differential Equations

"We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a blackbox differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum likelihood, without partitioning or ordering the data dimensions. For training, we show how to scalably backpropagate through any ODE solver, without access to its internal operations. This allows end-to-end training of ODEs within larger models."

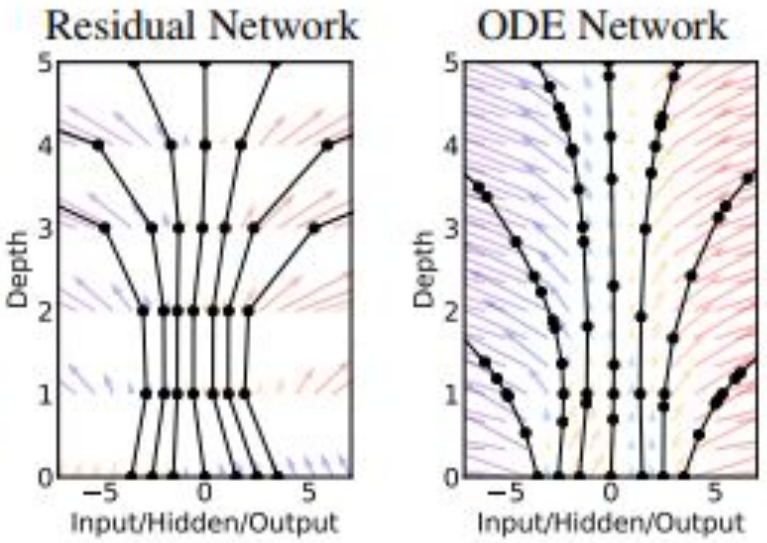
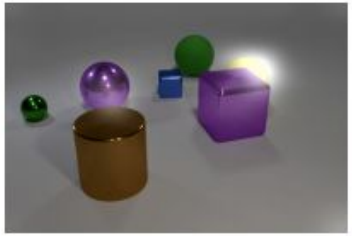
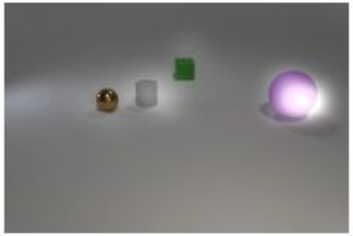


Figure 1: *Left:* A Residual network defines a discrete sequence of finite transformations. *Right:* A ODE network defines a vector field, which continuously transforms the state. *Both:* Circles represent evaluation locations.

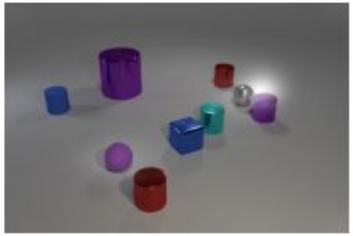
Memory, Attention, Composition Net



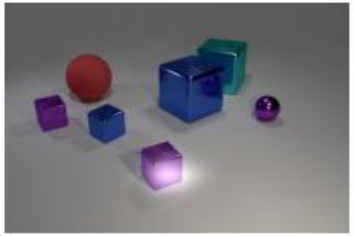
Q: *What is the shape of the large item, **mostly occluded** by the metallic cube?* **A:** *sphere* ✓



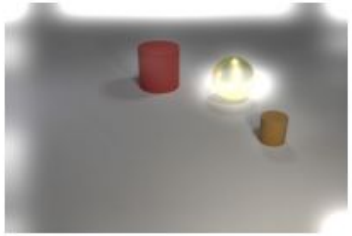
Q: *What color is the object that is a **different** size?* **A:** *purple* ✓



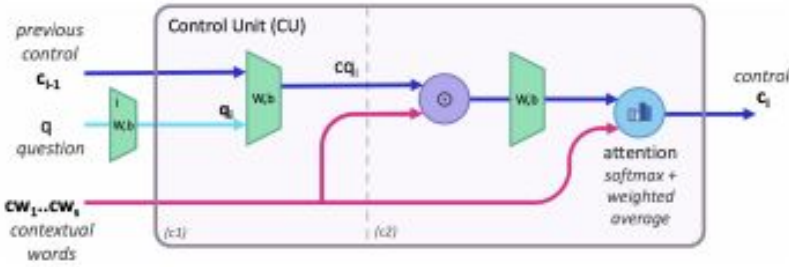
Q: *What color ball is **close to** the small purple cylinder?* **A:** *gray* ✓



Q: *What color block is **farthest front**?* **A:** *purple* ✓



Q: *Are any objects **gold**?* **A**: *yes* ✓

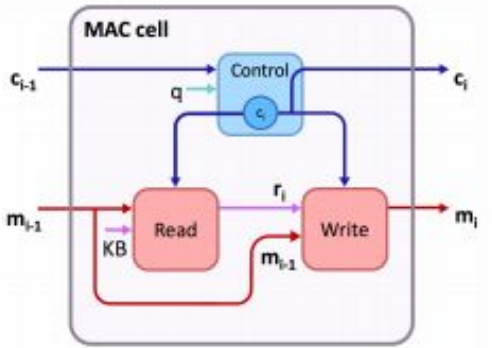


$$cq_i = W^{d\times 2d}\,[c_{i-1}, q_i] + b^d$$

$$ca_{i,s} = W^{1\times d}(cq_i \odot cw_s) + b^1$$

$$cv_{i,s} = \text{softmax}(ca_{i,s})$$

$$c_i = \sum_{s=1}^{S} cv_{i,s} \cdot cw_s$$



CycleGan

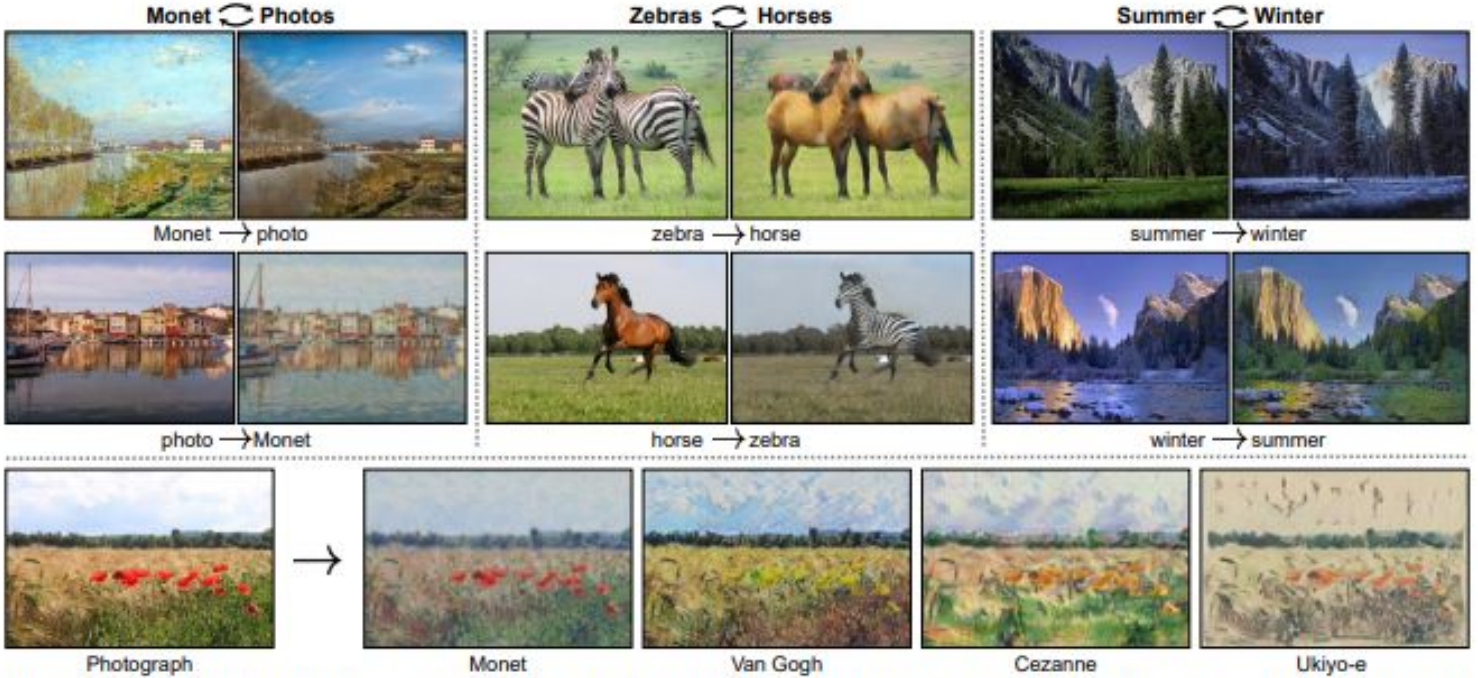
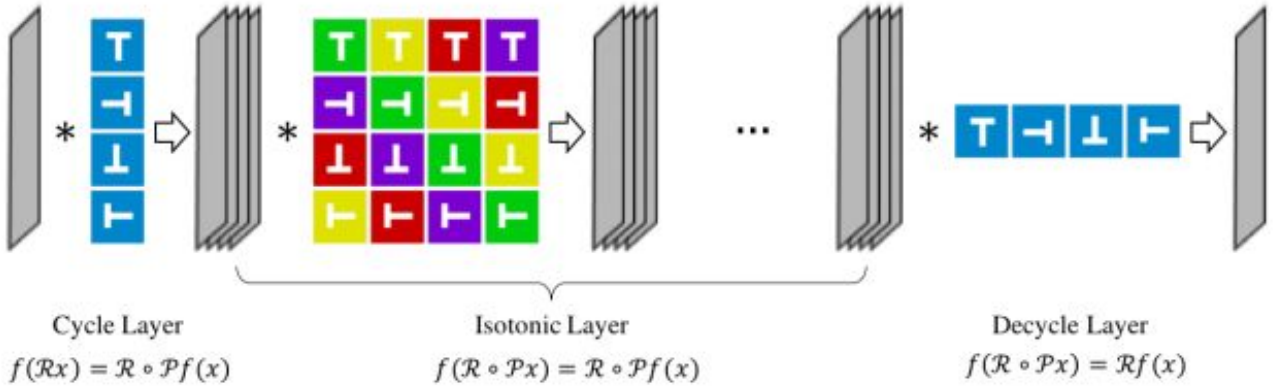


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

Rotation invariant convolutions

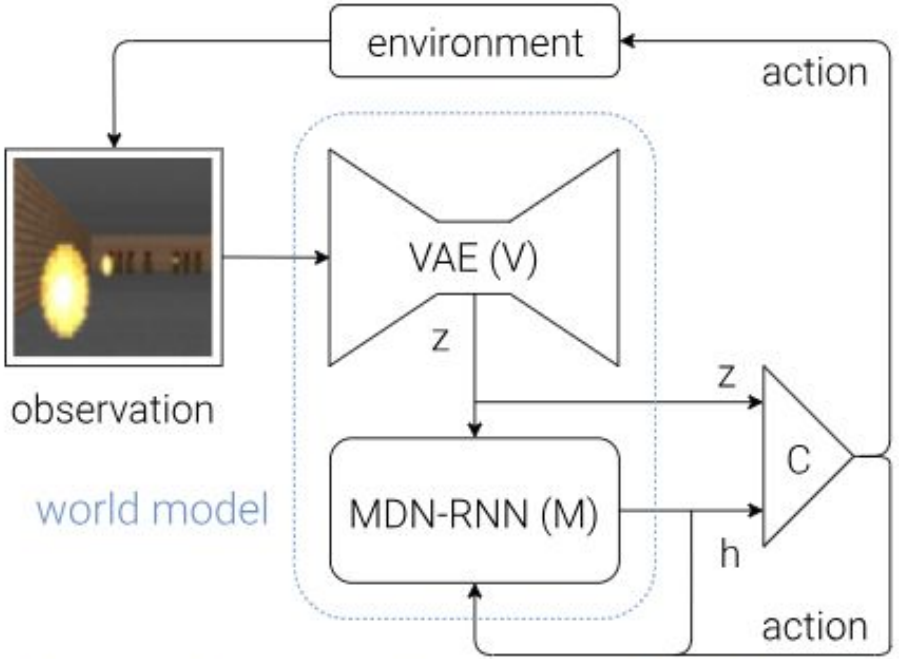
- Spatial transformer networks



World models

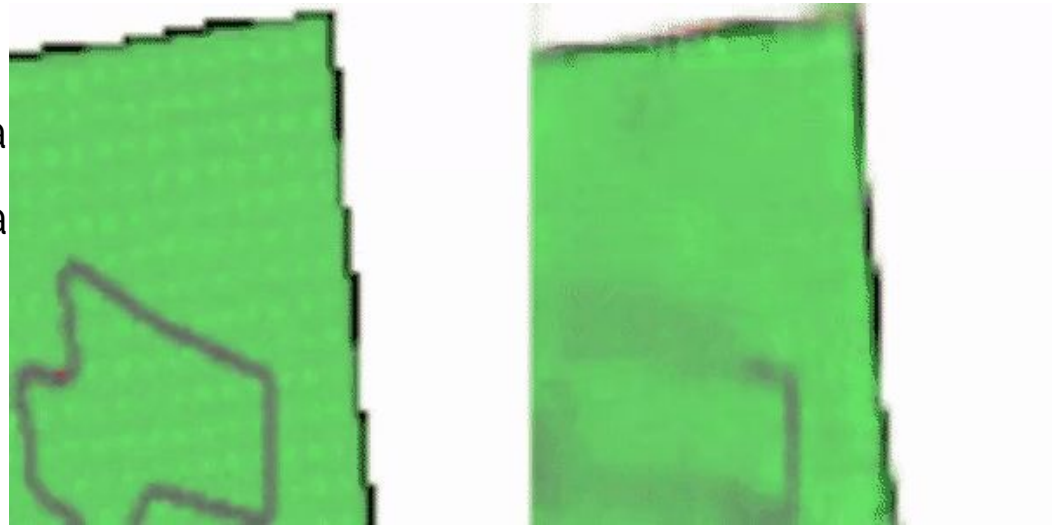
La observación se procesa por el VAE para generar el vector latente **z.**

Este se da como input al controlador, concatenado con el estado latente del modelo M en cada time step.



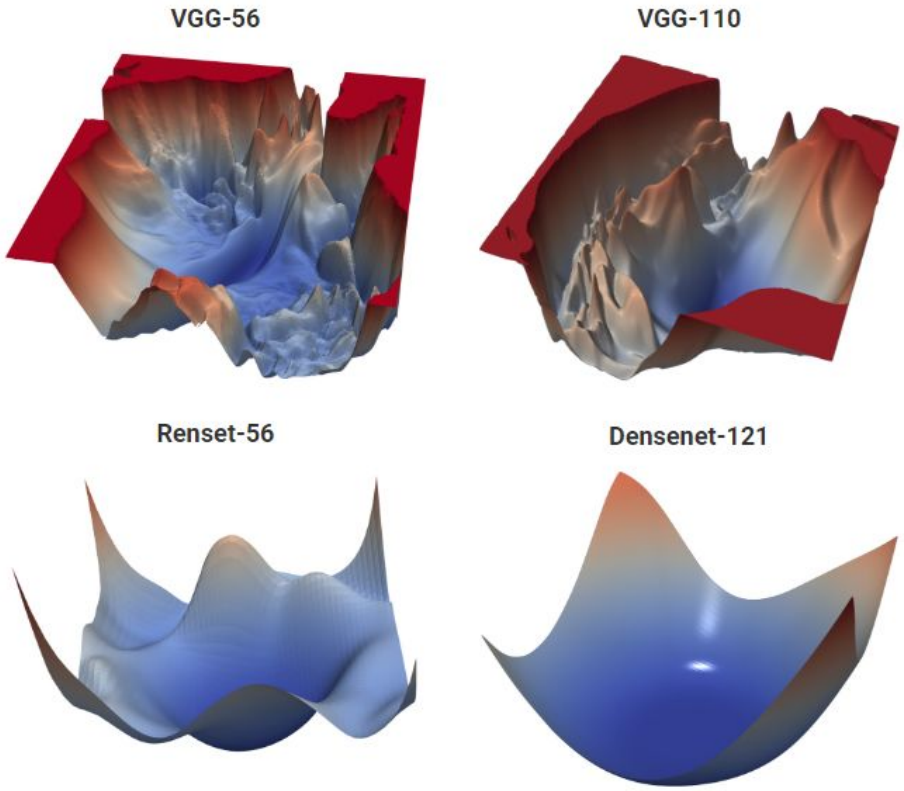
World models

- Recolectar 10000 observaciones con política random
- Entrenar VAE con esos frames $z \in R32$.
- Entrenar MDN-RNN para modelar $P(zt+1 \mid at, zt, ht)$
- Desarrollar el Controller para maximizar el refuerzo espera de un episodio



Teoría de deep learning

- Teoría de la información, Dimensión VC
- Backprop as a functor
- Caracterizando el contorno de la función de pérdida



IA en el mundo real

IA en mercado laboral

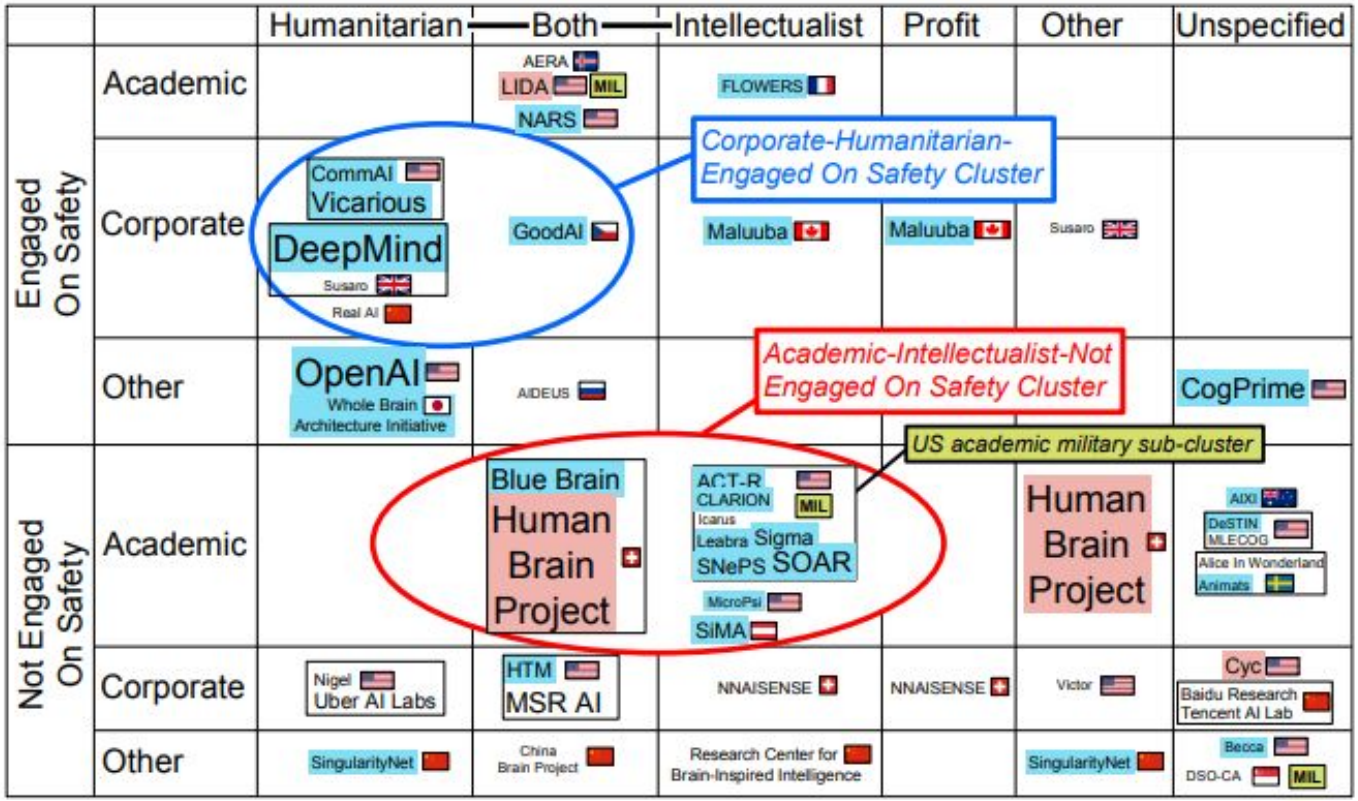
Ética

Ética

https://www.youtube.com/watch?v=9CO6M2HsoIA

https://mondoweiss.net/2017/07/orwell-arrest-algorithm/

Ética


Humanitarian
Both
Intellectualist
Profit
Other
Unspecified
Engaged On Safety
Not Engaged On Safety
Academic
Corporate
Other
AERA
LIDA
MIL
NARS
FLOWERS
CommAI
Vicarious
DeepMind
Susaro
Real AI
GoodAI
Maluuba
Maluuba
Susaro
OpenAI
Whole Brain Architecture Initiative
AIDEUS
CogPrime
Corporate-Humanitarian-Engaged On Safety Cluster
Academic-Intellectualist-Not Engaged On Safety Cluster
US academic military sub-cluster
Blue Brain
Human Brain Project
ACT-R
CLARION
MIL
Icarus
Leabra
Sigma
SNePS
SOAR
MicroPsi
SiMA
Human Brain Project
AIXI
DeSTIN
MLECOG
Alice In Wonderland
Animats
Nigel
Uber AI Labs
HTM
MSR AI
NNAISENSE
NNAISENSE
Victor
Cyc
Baidu Research
Tencent AI Lab
SingularityNet
China Brain Project
Research Center for Brain-Inspired Intelligence
SingularityNet
Becca
DSO-CA
MIL

Ética

- Well-being
- Respect for autonomy
- Protection of privacy and intimacy
- Solidarity
- Democratic participation
- Equity
- Diversity inclusion
- Prudence
- Responsibility
- Sustainable Development

< >
Montréal Declaration
Responsible AI_
</ >

Ética

- Bias
- Tracking
- Manipulación
- Automatización
- Privacidad
- Equidad

Próximos pasos

Tips

- Batch/Layer Normalization
- Cosine annealing/ΔLR
- Batch size
- No cerrarse a alternativas (tanh)
- Residuales

- TPUs Colab!
- Hyperparameters - AutoML, Adanet, Destilation
- Cómo presentar DS
- Roles en DS

Repaso

- Data Tabular/Shalloe
- NLP
- Secuencias
- Audio
- Imágenes

- Reinforcement
- Genetic
- Deploy
- Mercado

Extras: Executive Guide to AI

https://colab.research.google.com/drive/1AsVS4pFXIba38-XxQ1IIS6Eq-h5Tf3Ap

The End