Ciencia de Datos

2024

Introducción a la Ciencia de Datos

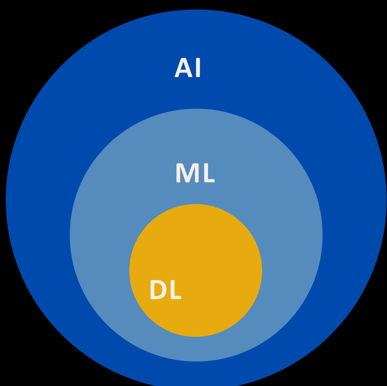
OPTATIVA - LICENCIATURA EN INFORMÁTICA
FACET-UNT

Ciencia de Datos: Fundamentos y Herramientas

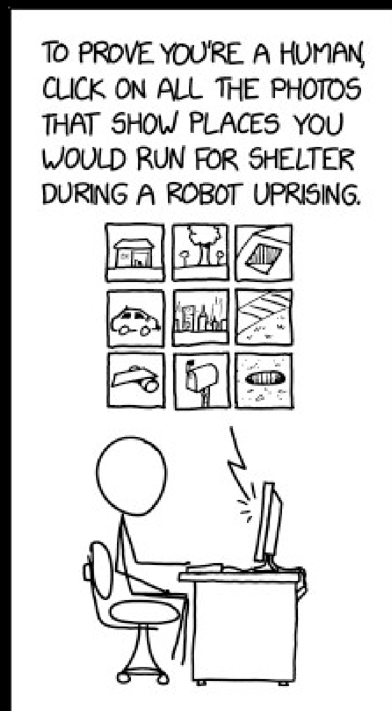
CURSO DE POSTGRADO - FACET-UNT

CD2023

Deep Learning



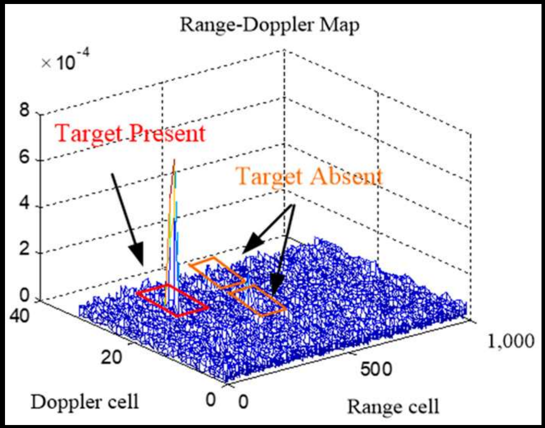
Extraer patrones de los datos usando redes neuronales

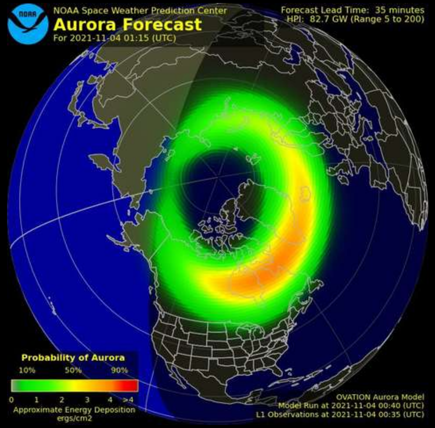


- https://this-person-does-not-exist.com/en

Recurrent Neural Networks

Redes neuronales recurrentes

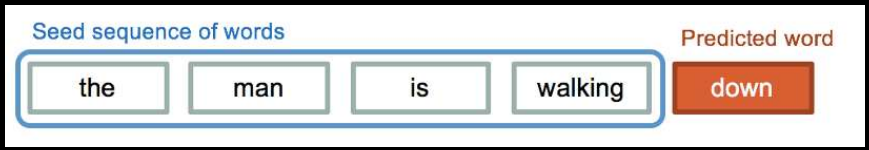


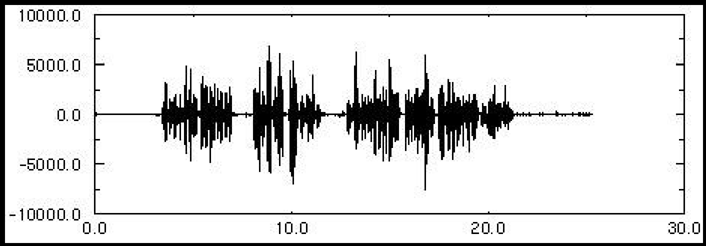


- Para procesar secuencias de datos x(t)= x(1), . . . , x(τ)

- Recurrente -> ejecura la misma tarea para cada elemento de la secuencia, la salida depende de varias cuentas realizadas en muestras previas

- Tiene “memoria”





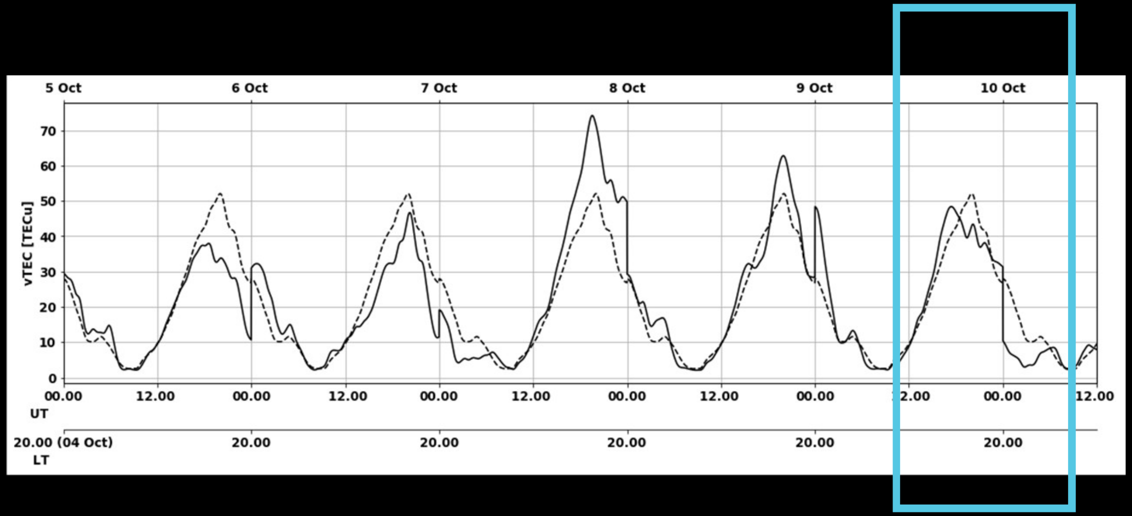
Recurrent Neural Networks

RNNs como una manera de modelar problemas de secuencias de datos

Queremos pronosticar esto!

Lo que necesitaria el modelo es:

- Manejar secuencias de **longitud variable**
- Trackear dependencias de **largo plazo (long-term** dependences)
- Mantener la informacion sobre el **orden**
- **Compartir parametros** a lo largo de la secuencia



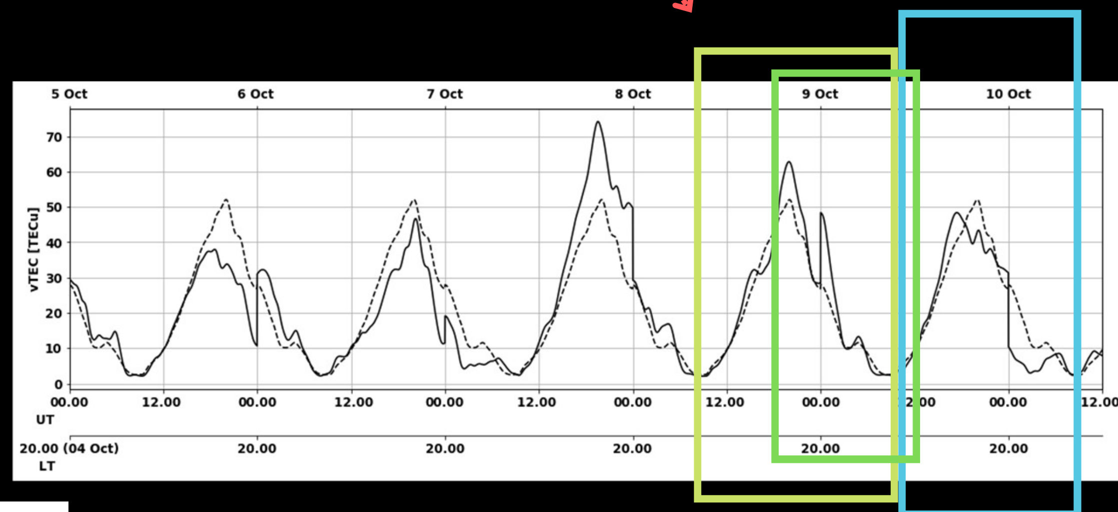
Recurrent Neural Networks

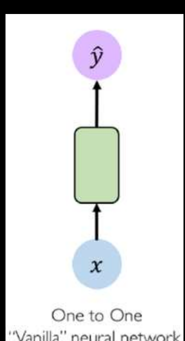
RNNs as an approach to sequence modeling problems

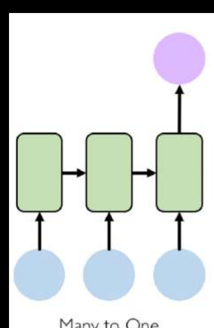
Lo que necesitaría el modelo es:

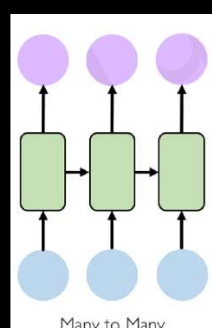
Cuantos pasos? (longitud)

- Manejar secuencias de **longitud variable**
- Trackear dependencias de **largo plazo (long-term** dependences)
- Mantener la informacion sobre el **orden**
- **Compartir parametros** a lo largo de la secuencia









Y + arquitecturas

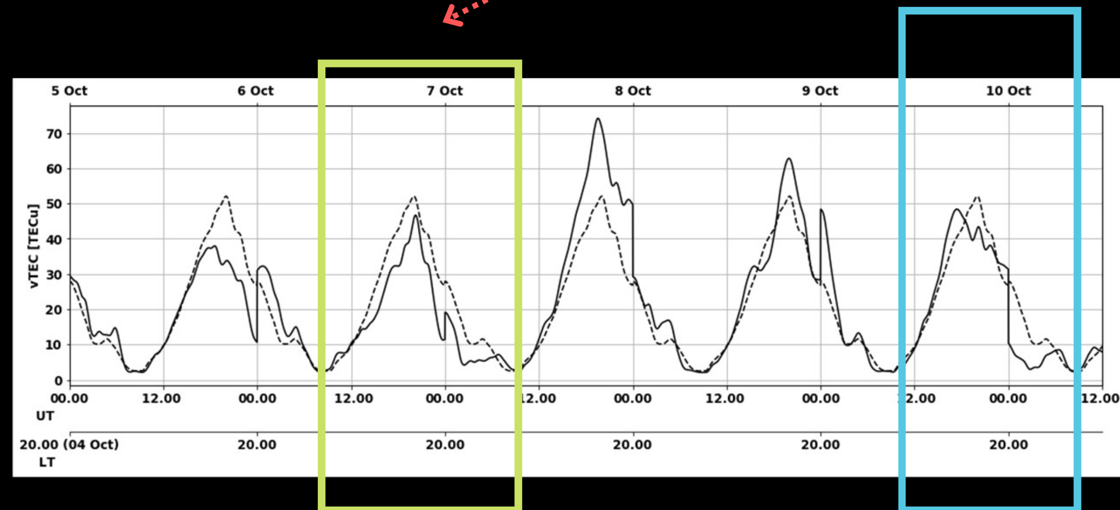
Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

Lo que necesitaria el modelo es:

- Manejar secuencias de **longitud variable**
- Trackear dependencias de **largo plazo (long-term** dependences)
- Mantener la informacion sobre el **orden**
- **Compartir parametros** a lo largo de la secuencia

Que tan importante es la informacion del pasado (lejano) ?



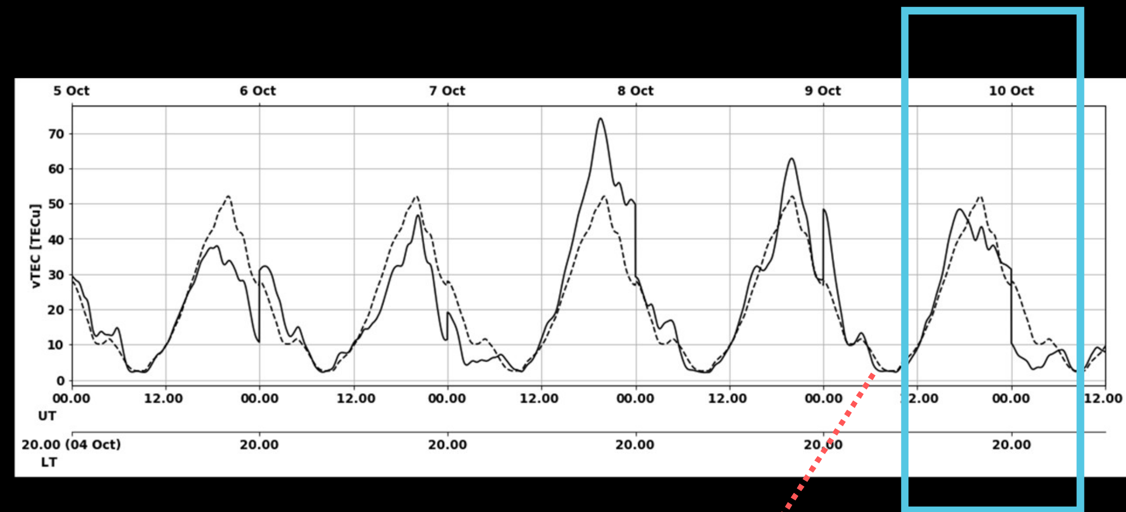
La serie podria tener comportamientos regulares (diarios, estacionales, etc.)

Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

Lo que necesitaria el modelo es:

- Manejar secuencias de **longitud variable**
- Trackear dependencias de **largo plazo (long-term** dependences)
- Mantener la informacion sobre el **orden**
- **Compartir parametros** a lo largo de la secuencia



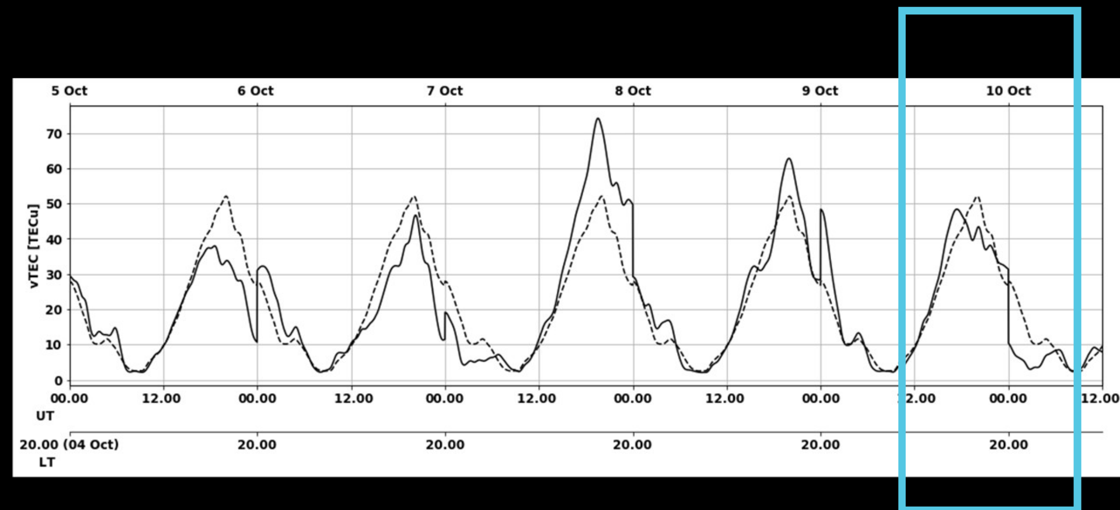
vTEC (t-2), vTEC (t-1), vTEC (t-0) <> vTEC (t-0), vTEC (t-2), vTEC (t-1)

Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

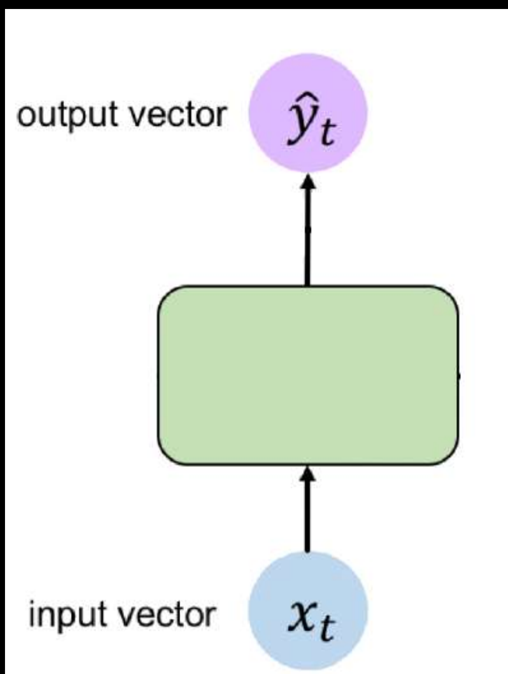
Lo que necesitaría el modelo es:

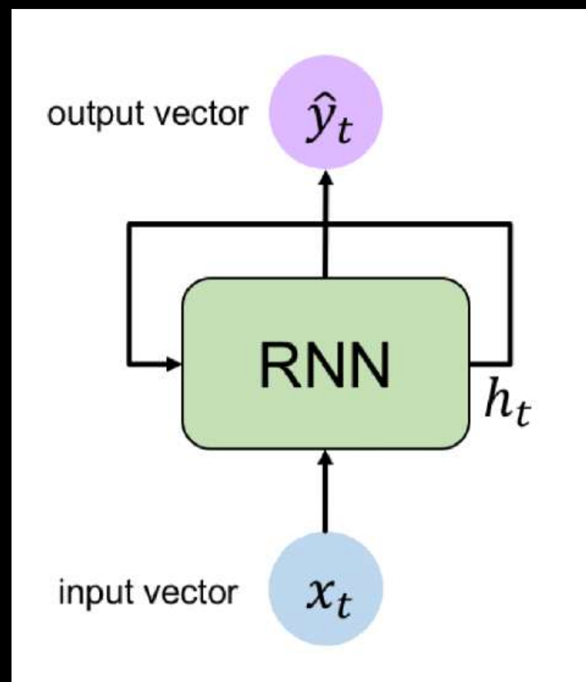
- Manejar secuencias de **longitud variable**
- Trackear dependencias de **largo plazo (long-term dependences)**
- Mantener la informacion sobre el **orden**
- **Compartir parametros** a lo largo de la secuencia



RNNs tienen un estado o state (h_t), que es actualizado en cada paso a medida que la secuencia es procesada usando los mismos parametros en cada paso de tiempo

Recurrent Neural Networks



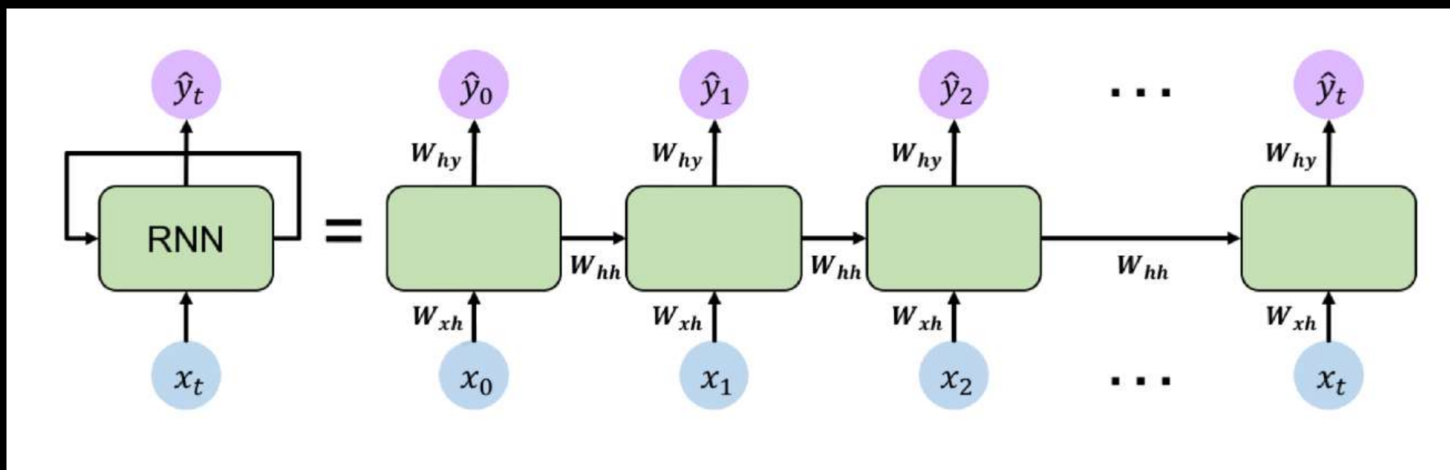


$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

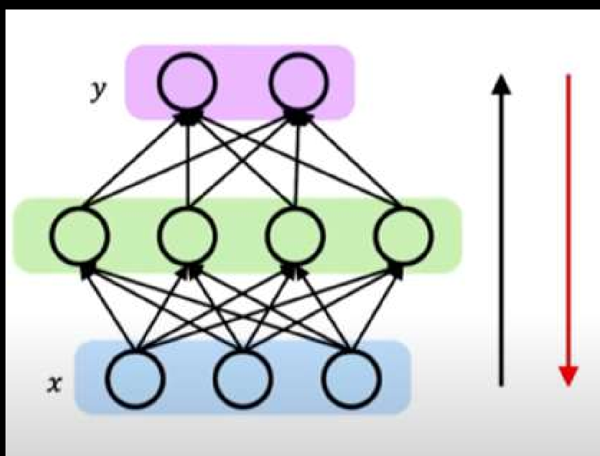
cell state | function parameterized by W | old state | input vector at time step t

- Aplica una **relacion recurrente** en cada paso de tiempo para procesar la secuencia

Recurrent Neural Networks



Backpropagation through time: long time dependences



$$\boxed{\begin{gathered} W^* = \underset{W}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; W\right), y^{(i)}\right) \\ W^* = \underset{W}{\operatorname{argmin}} J(W) \end{gathered}}$$

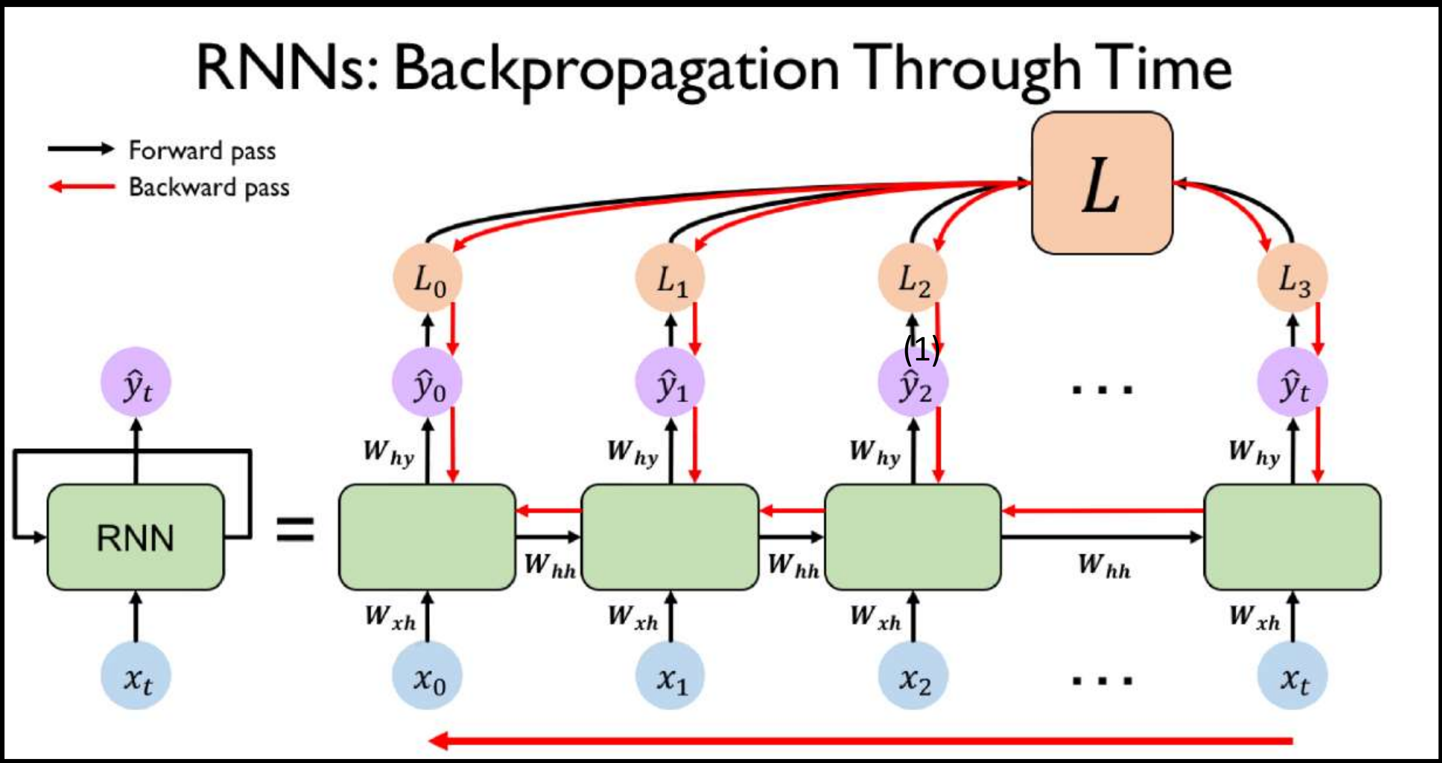
$$\downarrow$$

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(W)}{\partial W}$
4. Update weights, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
5. Return weights

- Toma el gradiente de la perdida (loss) con respecto a cada parametro
- Cambia parametros para poder minimizar la perdida

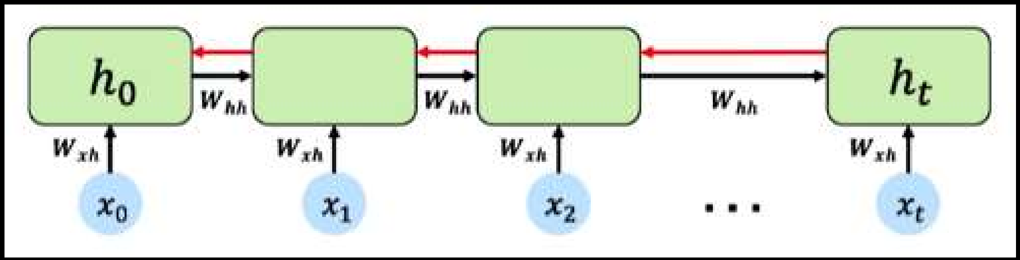
Backpropagation through time: long time dependences



- Realiza la computacion individual de l_i para cada paso y los suma
- Hace backpropagation de los errores individuales de cada paso y luego de todos los pasos de tiempo hasta el inicio de la secuencia

https : //kharshit.github.io/blog /2019/02/22/backpropagation – through – time

Backpropagation through time: long time dependences



- Mucho tiempo de computo!

- Muchos valores >>1 -> exploding gradient (*)
- Muchos valores <<1 -> vanishing gradient

(*) Gradient clipping is a simple technique: If the gradient gets too large, we rescale it to keep it small.

Vanishing gradient problem

Multiplica muchos numeros pequenos

Errores debido a que para aquellos pasos lejanos en el tiempo, los gradientes se hacen cada vez mas pequenos

Sesga los parametros y captura mayormente las dependencias de corto plazo

Como resolver el problema:

- Activation function
- Inicializacion de pesos
- Nuevas arquitecturas

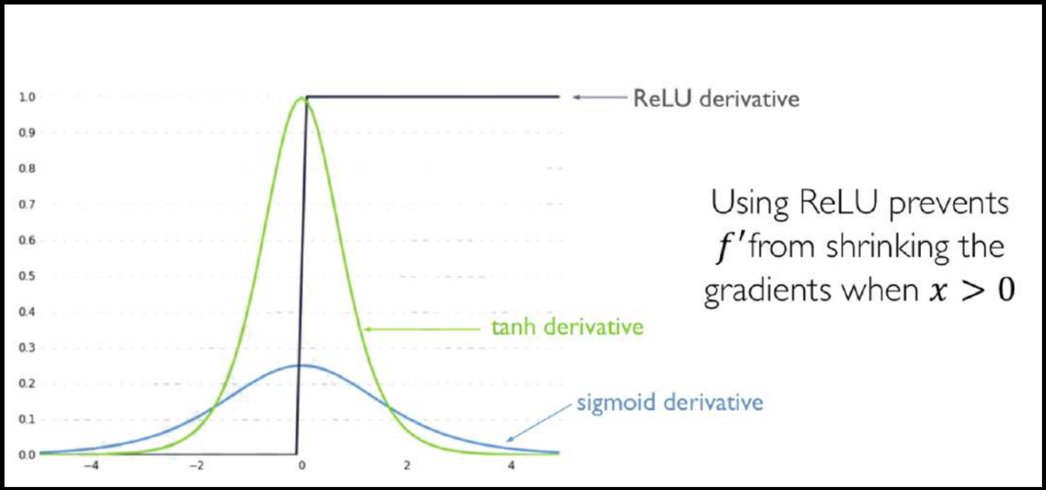
Vanishing gradient problem

Multiplica muchos numeros pequenos

Errores debido a que para aquellos pasos lejanos en el tiempo, los gradientes se hacen cada vez mas pequenos

Sesga los parametros y captura mayormente las dependencias de corto plazo

- Activation function



Vanishing gradient problem

Multiplica muchos numeros pequenos

Errores debido a que para aquellos pasos lejanos en el tiempo, los gradientes se hacen cada vez mas pequenos

Sesga los parametros y captura mayormente las dependencias de corto plazo

Inicializacion de pesos

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Inicializar weigths con la matriz identidad
Initializar biases com cerp

Previene que los gradientes se vayan a cero

Vanishing gradient problem

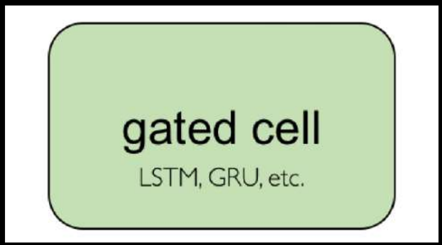
Multiplica muchos numeros pequenos

Errores debido a que para aquellos pasos lejanos en el tiempo, los gradientes se hacen cada vez mas pequenos

Sesga los parametros y captura mayormente las dependencias de corto plazo

- Nuecas arquitecturas

Solucion mas robusta

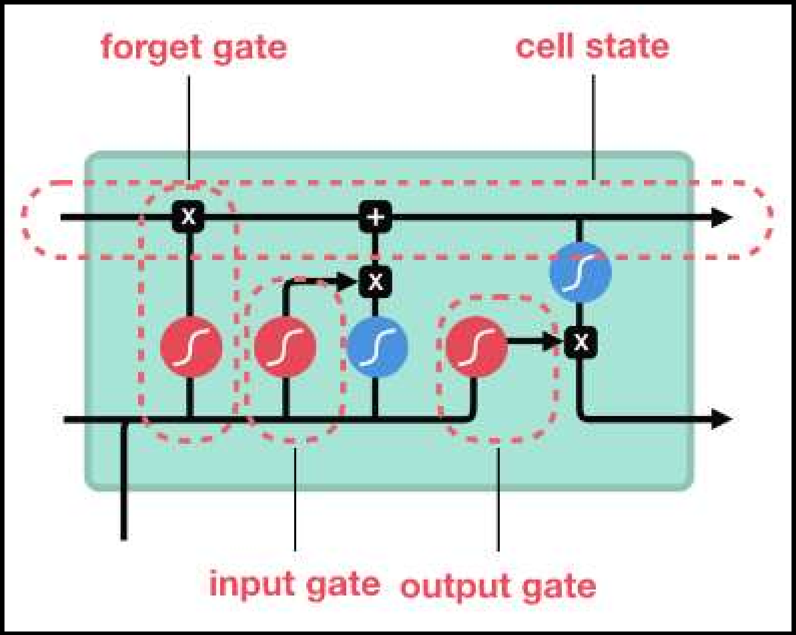


Usa unidades recurrentes con compuertas que son mas complejas para controlar el flujo de la informacion que se esta pasando a lo largo de la secuencia

Las compuertas agregan o remueven informacion selectivamente dentro de cada unidad recurrente

Long short term memory (LSTM)

Gates: 1) Forget 2) Input (store) 3)Update 4) Output



Como trabaja:

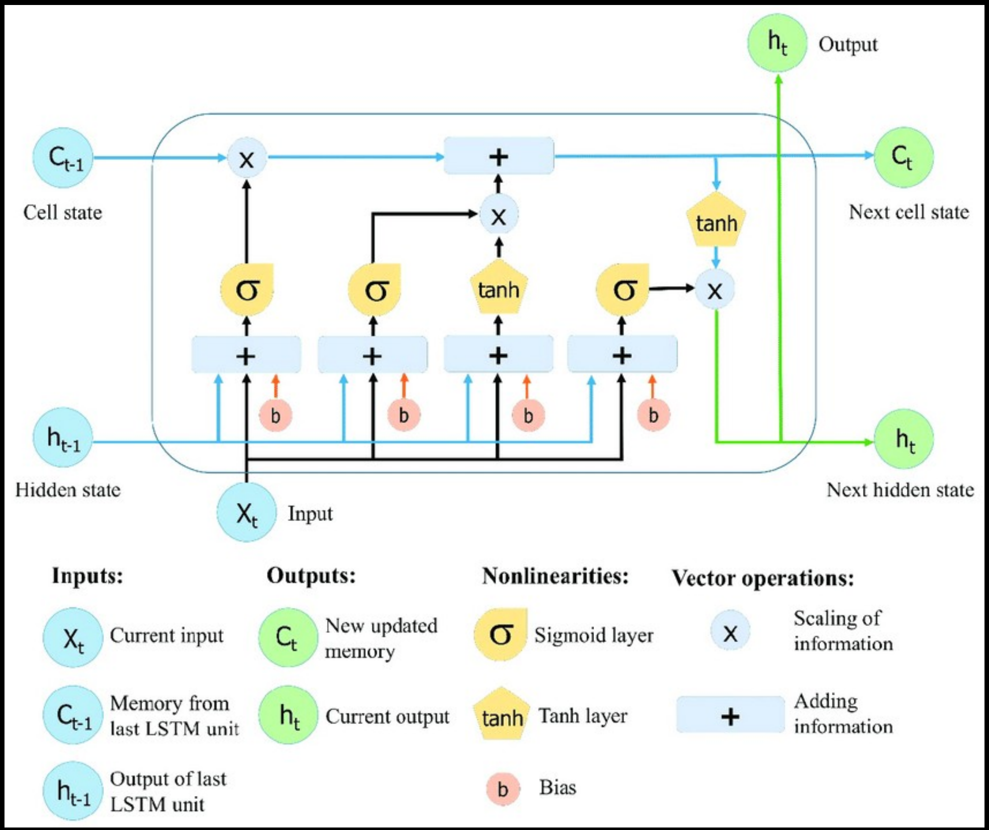
1)Mantiene una celda de estado
2)Usa compuertas (gates) para controlar el flujo de la informacion.

- Forget gate: se deshace de la informacion poco relevante
- Almacien informacion relevante de la entrada actual (store/input gate)
- Selectivamente hace un update de la celda de estado

Output gate: retorna la version filtrada de la celda de estado
3) Backpropagation TT con flujo de gradientes parcialmente sin interrupciones

Long short term memory (LSTM)

Gates: 1) Forget 2) Input (store) 3)Update 4) Output



How it works:

1)Maintain a cell state

2)Use gates to control the flow of information

- Forget gate gets rid of irrelevant information
- Store relevant information from the current input
- Selectively update cell state
- Output gate returns a filtered version of the cell state

3) Backpropagation TT with partially uninterrupted gradient flow

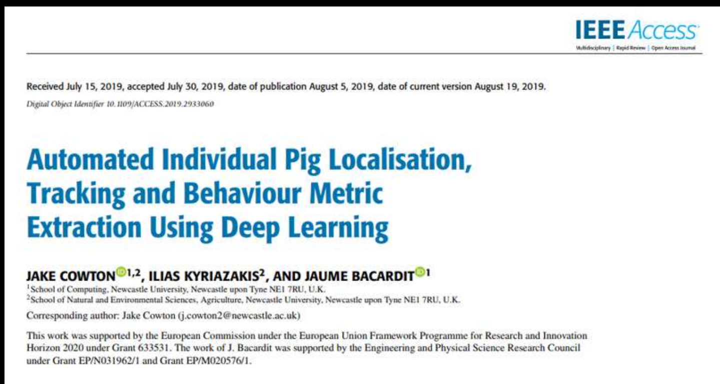
LSTM "no todo lo que brilla es oro"

- << vanishing gradient problem …. pero no lo elimina completamente.
- >> Costo computacional
- Es afectado por diferentes inicializaciones aleatoriaas de los pesos
- Drop-out es dificil de implementar
- Tiende a hacer overfitting
- << performance en problemas con dependencias temporales muy largas

Convolutional Neural Networks

CNNs

Computer Vision.
- Facial detection and recognition
- Healthcare, medicine and biology
- Self-driving vehicles


IEEE Access

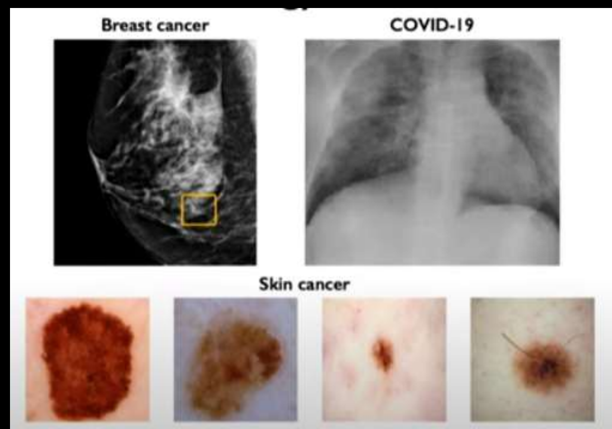
Received July 15, 2019, accepted July 30, 2019, date of publication August 5, 2019, date of current version August 19, 2019.
Digital Object Identifier 10.1109/ACCESS.2019.2933060

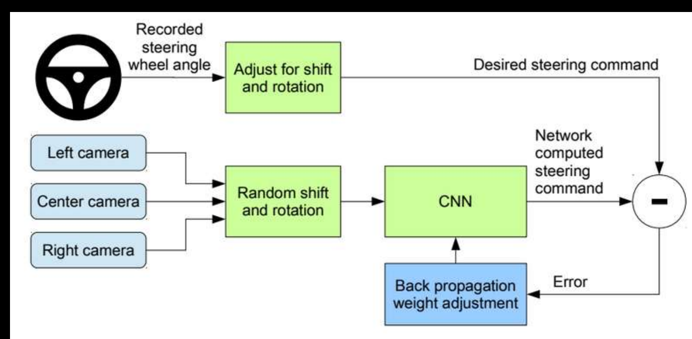
Automated Individual Pig Localisation, Tracking and Behaviour Metric Extraction Using Deep Learning

JAKE COWTON[1,2], ILIAS KYRIAZAKIS[2], AND JAUME BACARDIT[1]
[1]School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.
[2]School of Natural and Environmental Sciences, Agriculture, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.
Corresponding author: Jake Cowton (j.cowton2@newcastle.ac.uk)

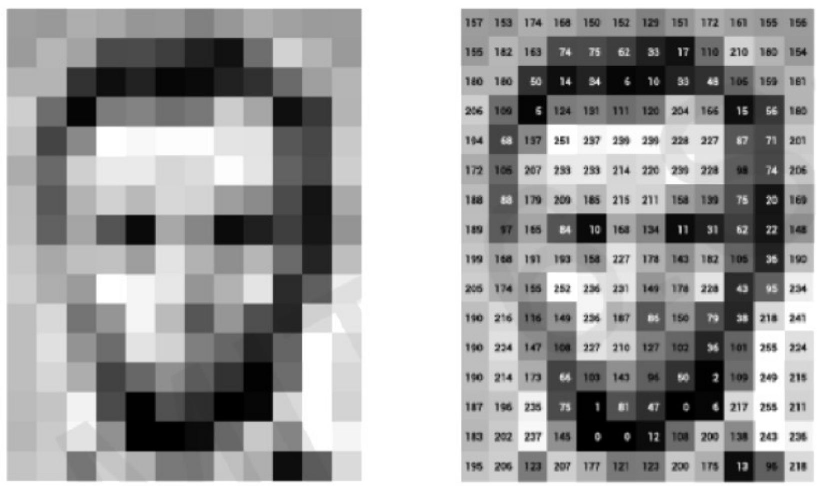
This work was supported by the European Commission under the European Union Framework Programme for Research and Innovation Horizon 2020 under Grant 633531. The work of J. Bacardit was supported by the Engineering and Physical Science Research Council under Grant EP/N031962/1 and Grant EP/M020576/1.

FIGURE 8. Four sample images from our pig detection test set processed by the Faster R-CNN with the feature extraction layers pre-trained on ImageNet, the rest pre-trained on Pascal Visual Object Classes Challenge 2007 and an additional fully-connected layer for the pig dataset. Detections to the left of the red wall are ignored. The top left image is from the low-light test segment. The top right image is from the densely packed test-segment. The bottom left image is from the overexposed test segment. The bottom right image is from the many pigs test segment.





Convolutional Neural Networks



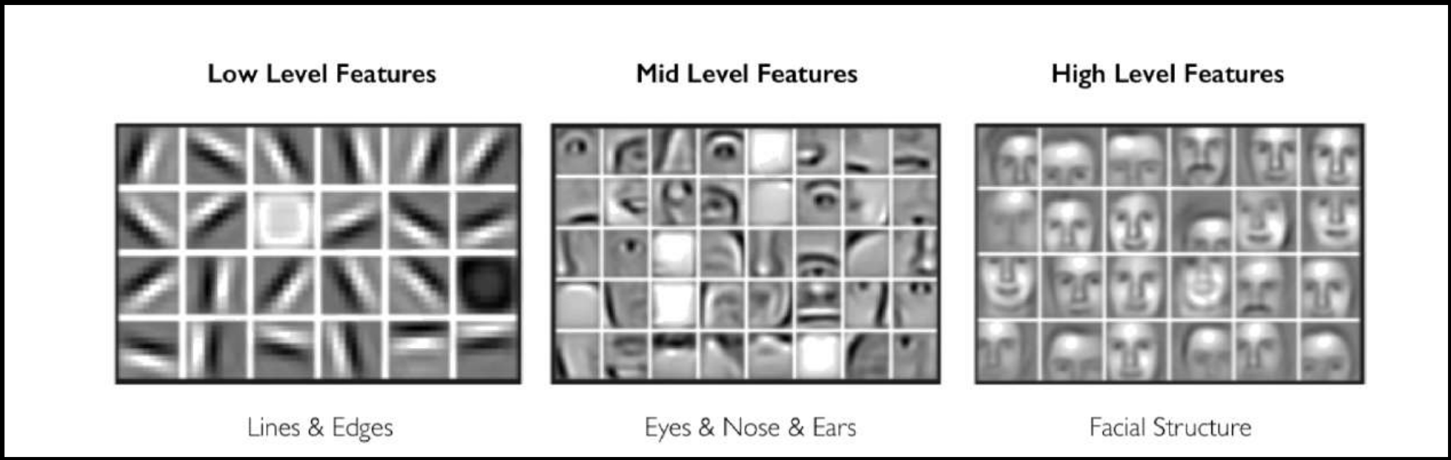
What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

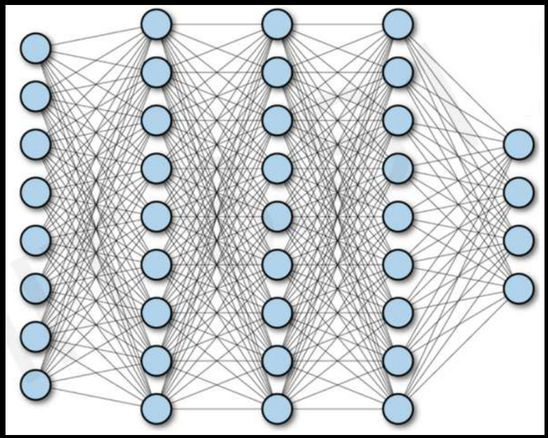
Features

(Como en otros problemas con NN)
- Regresion
- Classificacion
- Deteccion de features Ede alto nivel



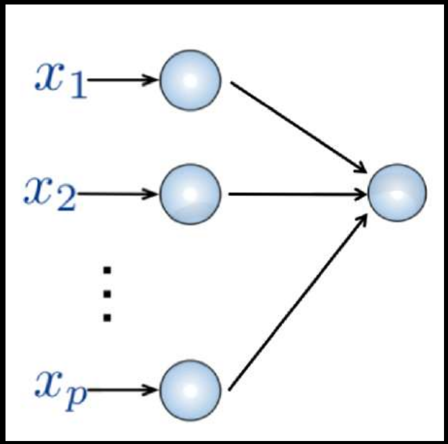
Extraer las features a partir de los datos!
Aprender jerarquias de features!

Fully connceted NN



Input:
- 2D image
- vector de valores de pixeles

(flatten the image)

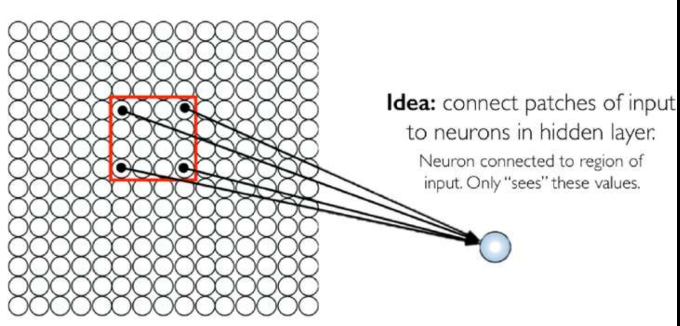


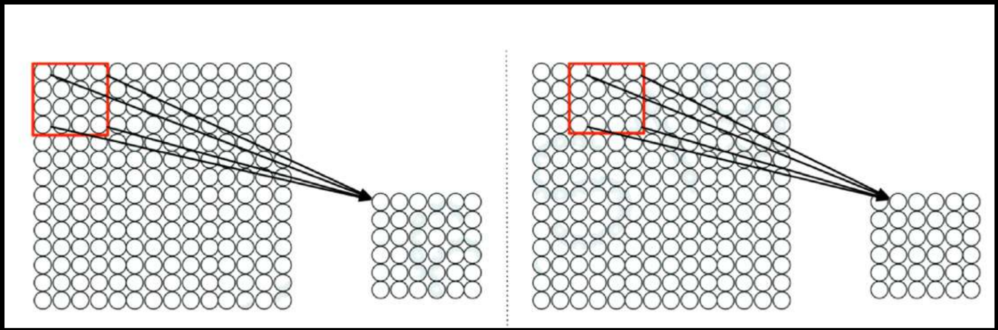
Fully connected:
- Conectar una neurona de una capa oculta a todas las neuronas en la entrada
- No mantiene informacion espacial!
- Muchos parametros

Como le agregamos una estructura esacial a la entrada?

Using spatial structure

Input:
- 2D image
- Arreglo de pixeles

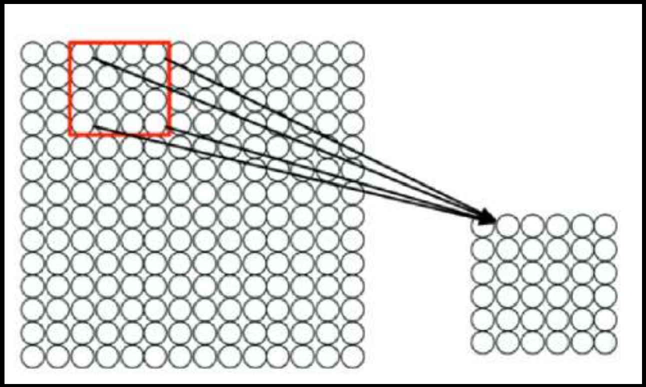




Sliding window para definir las conecciones (conecta un parche en la capa de entrada a una sola neurona)

La clave: como podemos pesar cada parche para detectar una feature particular?

Using spatial structure

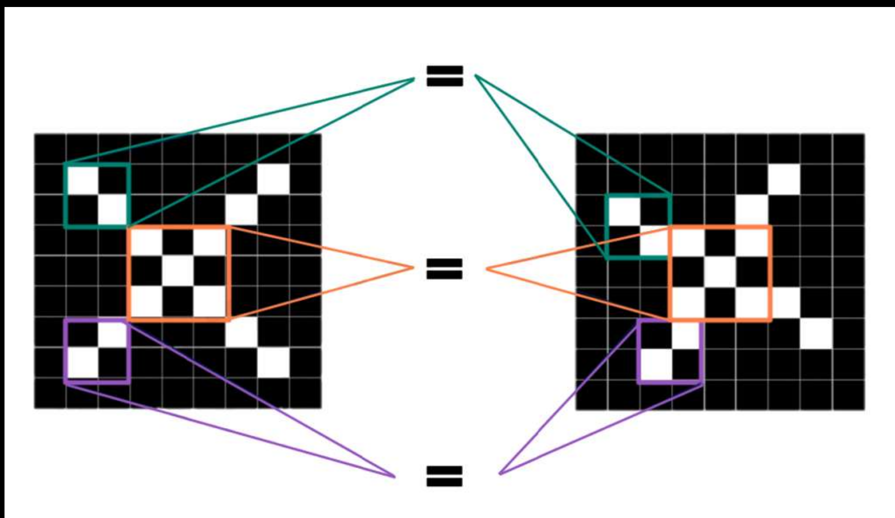


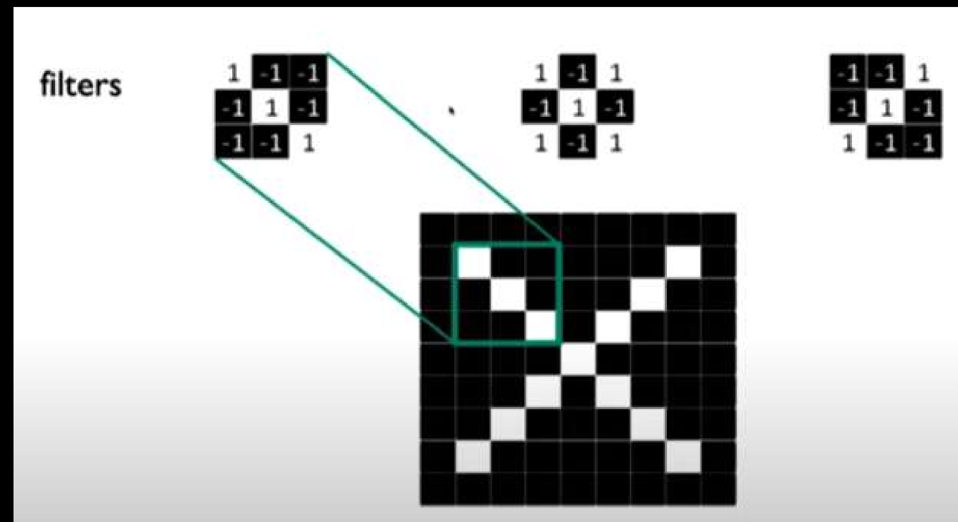
- Filtro de size 4x4: son 16 pesos diferentes
- Aplicar el mismo filtro a 4x4 parches en la entrada
- Mocer 2 pixels para el proximo parche

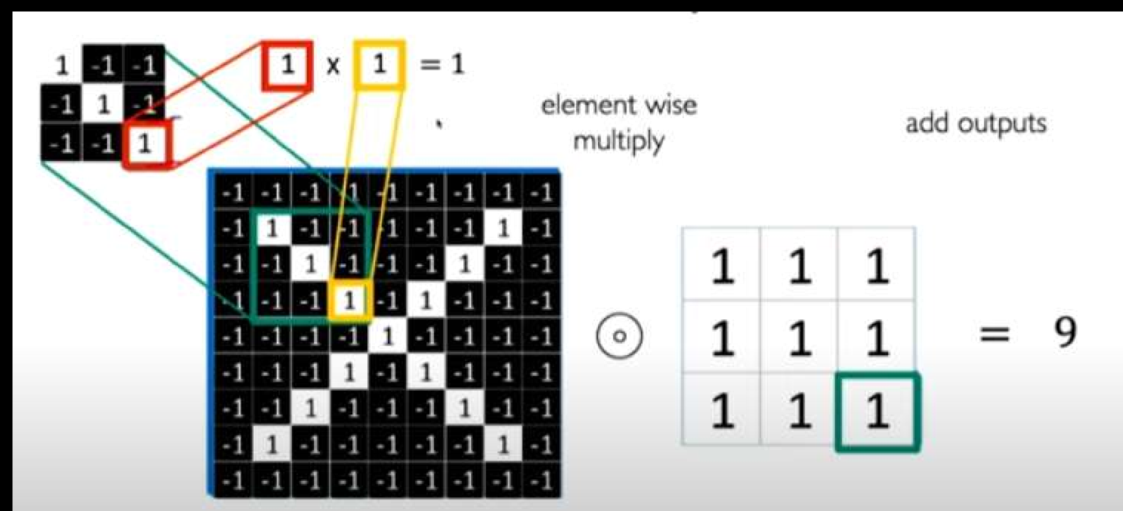
Esta operacion se llama convolucion

- Aplicar un set de pesos (un filtro) para extraer features locales
- Usar multiples filtros para extraer diferentes features
- Cada filtro comparte espacialmente parametros

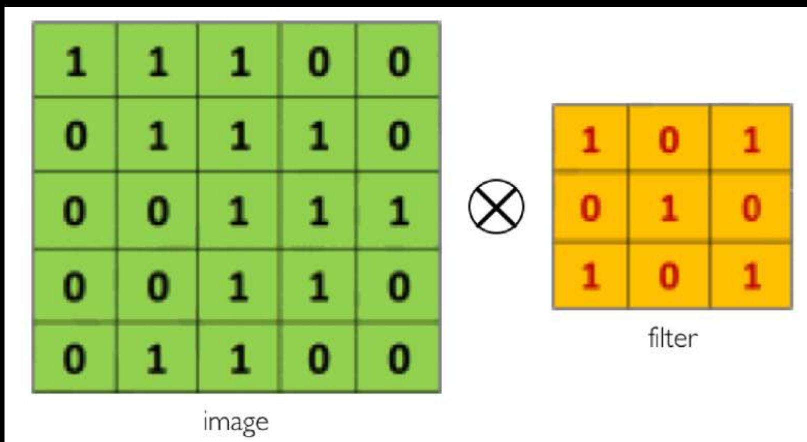
Convolutional operation

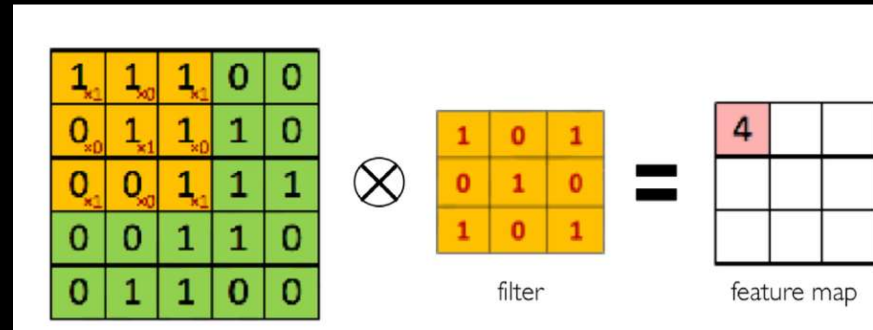


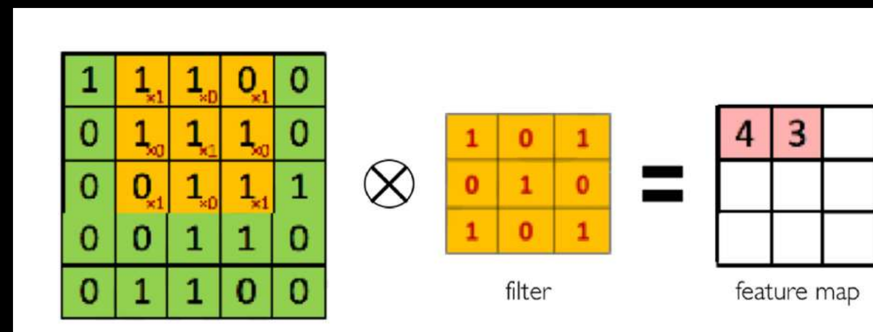




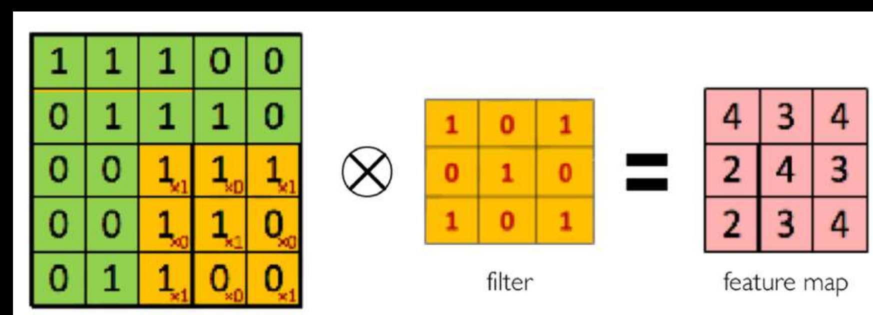
Convolutional operation





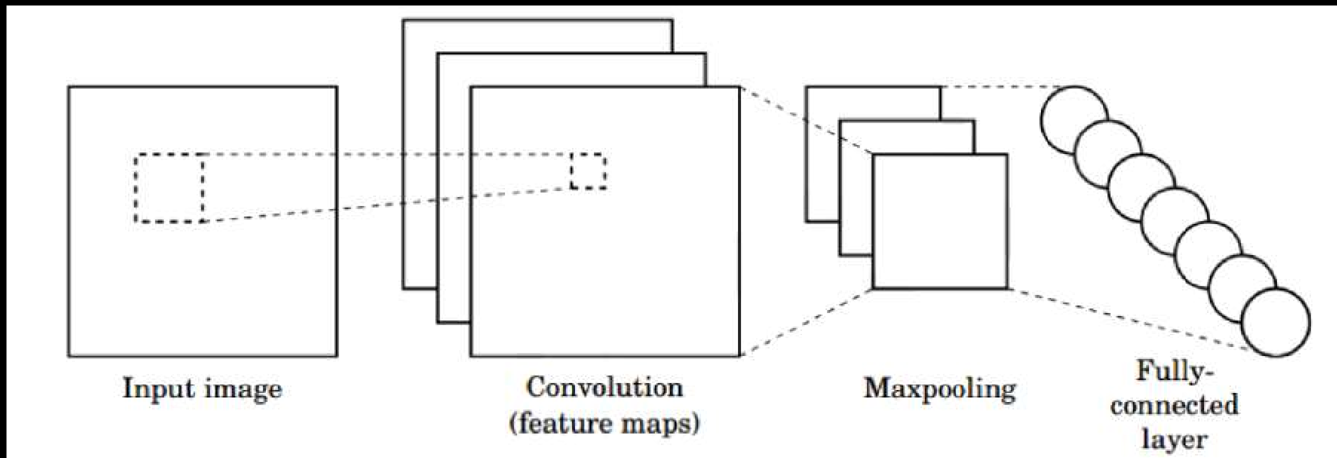


• • •



Convolutional neural networks (CNN)

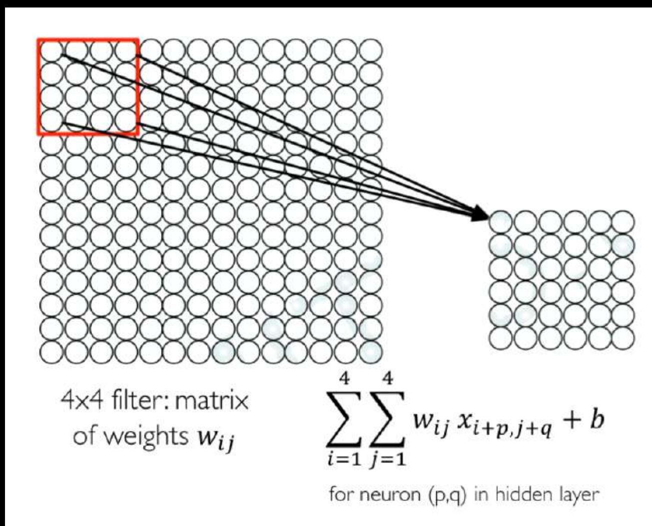
- for classification

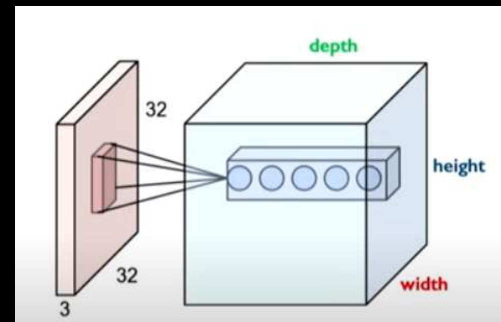


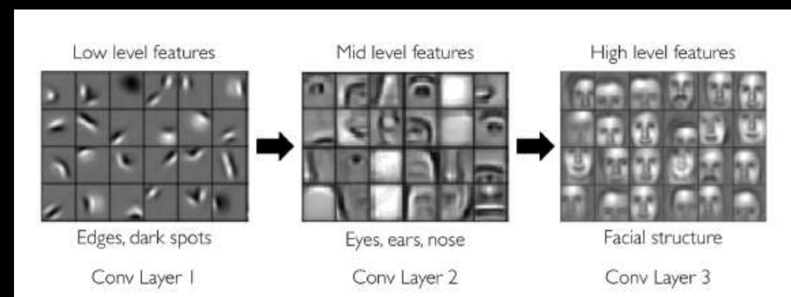
- Convolucion: aplicar filtros para generar’feature maps’
- Non-linealidad: mas usado ReLU
- Pooling: operacion de submuestreo (downsampling) en cada mapa de features

Train model with image data.
Learn weights of filters in convolutional layers.

Convolutional layer







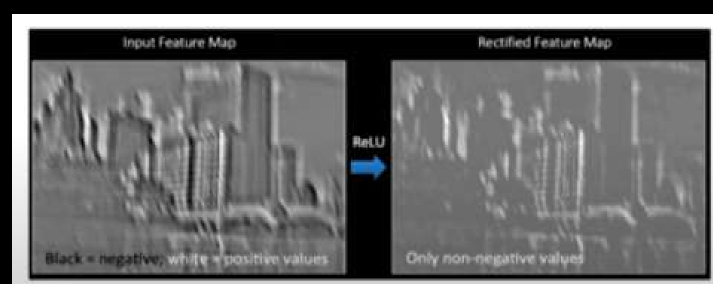
1. Aplicar una ventana de pesos

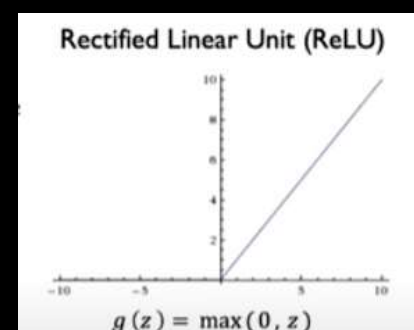
2. calcular las combinaciones lineales

3. se activa con una funcion no lineal

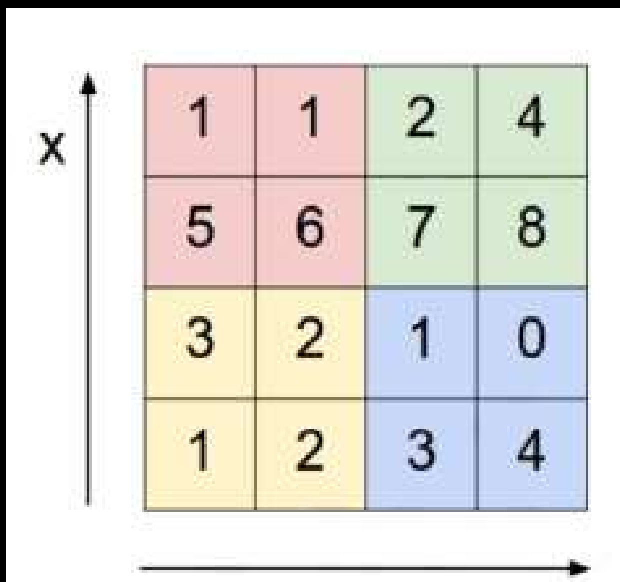
Para cada neurona de la capa oculta:

- Toma las entradas del parche
- Calcula la suma ponderada
- Aplica bias





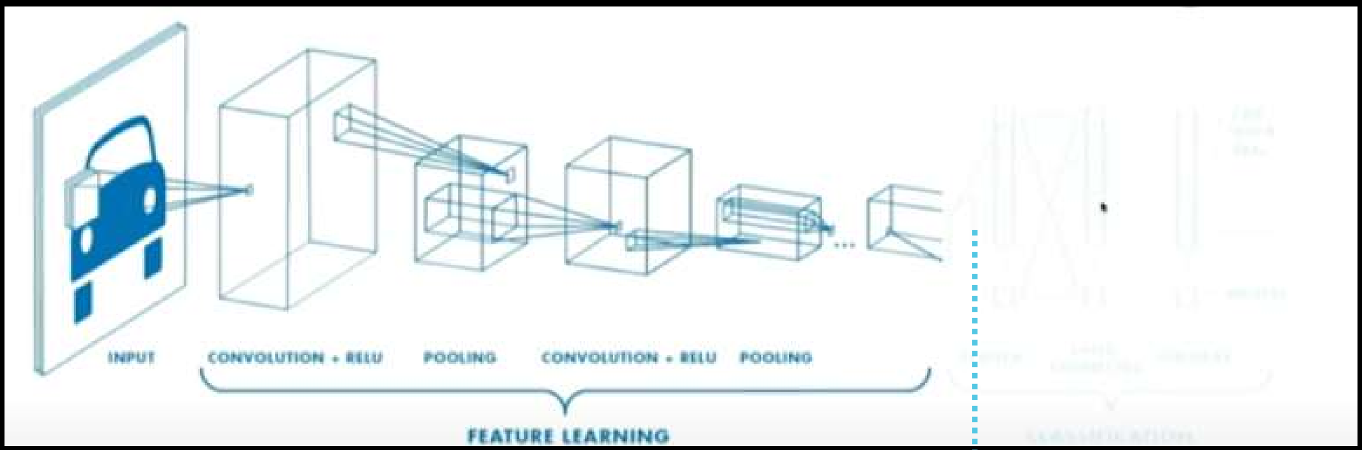
Pooling



max pool con filtro 2x2
Y un paso de 2

- Reduce la dimensionalidad
- Invarianza espacial

CNN: classification example



- Hasta este punto, la ultima dalida e el i-esimo feature map

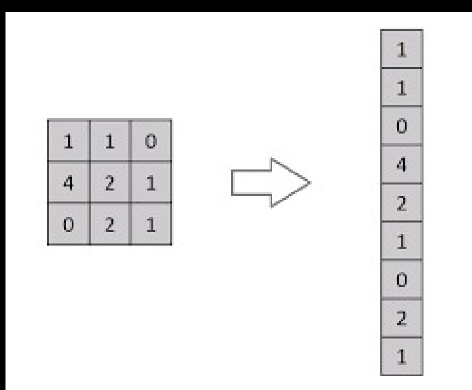
CNN: classification example

CONV and POOL layers putput high-level features input

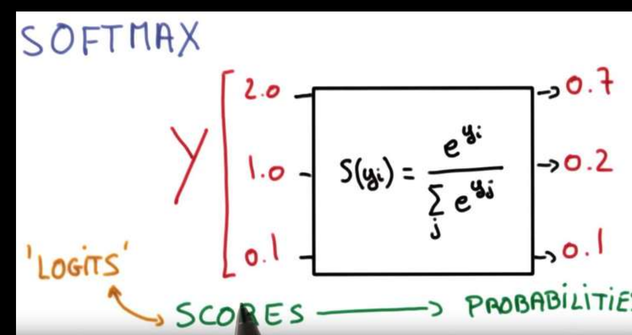
Saluda de las capas CONV y POOL >> features de alto nivel

Una capa fully connected usa estas features para clasificar la image

Expresa la salida como una probabilidad (de la imagen a una cierta clase)



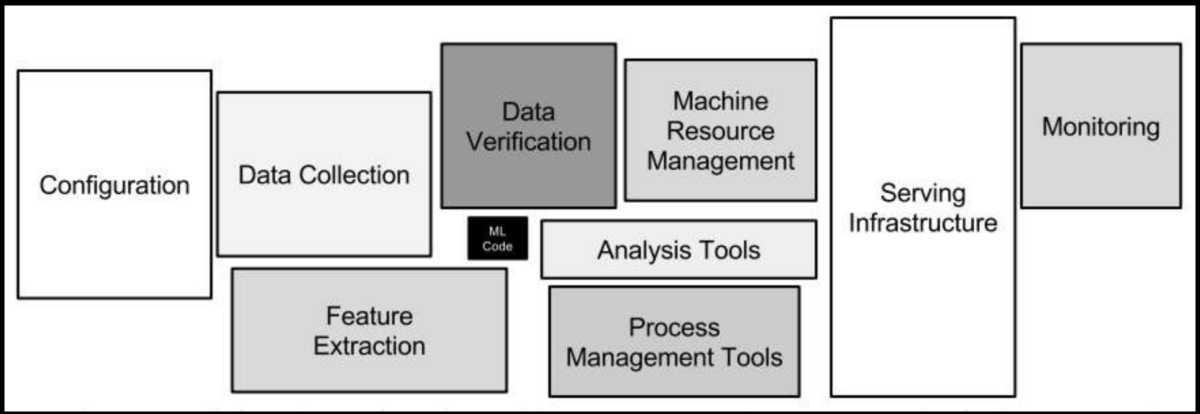
$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$



logits = predicciones no normalizados (aun) del modelo


¿Y AHORA QUÉ?

Deployment



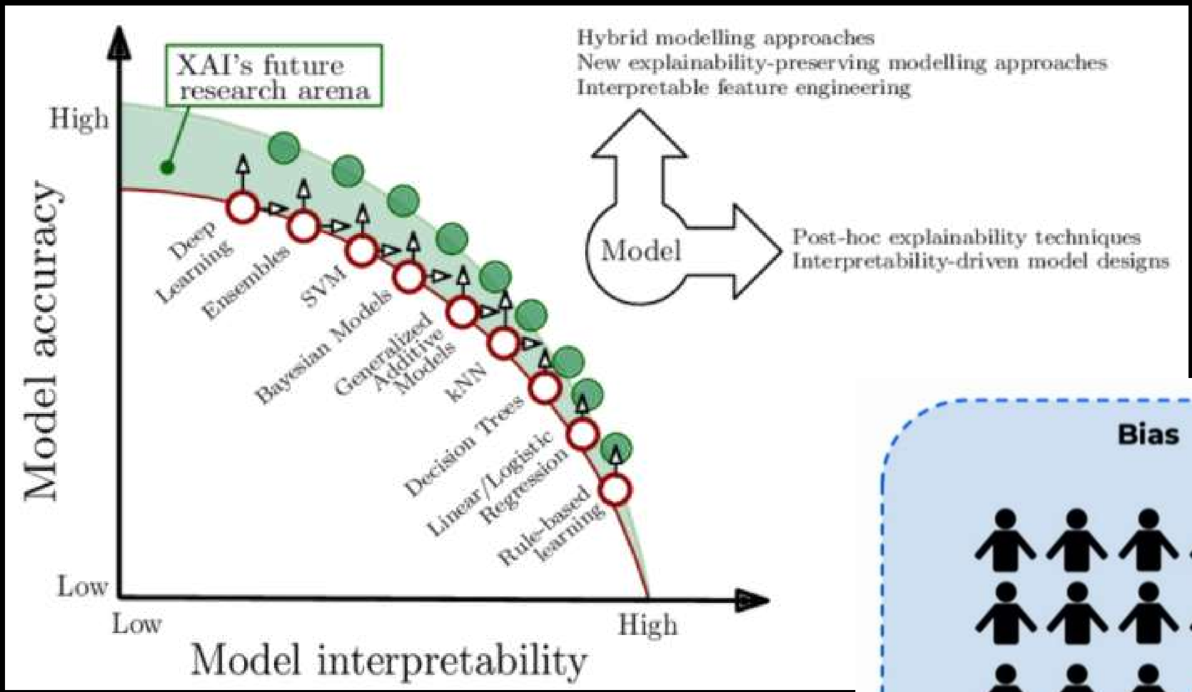
Hidden Technical Debt in Machine Learning Systems, D. Sculley et.al (2015)

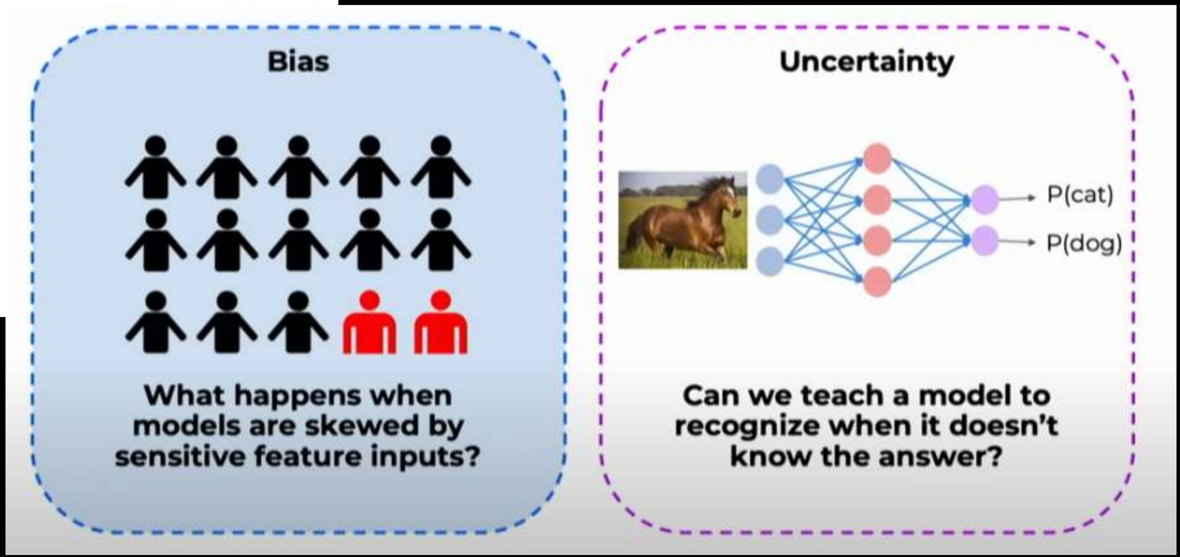
- Tiene todos los problemas de deployment y manenimiento de un codigo tradicional mas los problemas especificos de ML
- El modelo es solo una pequena parte de un sistema operacional

- How easily can an entirely new algorithmic approach be tested at full scale?
- What is the transitive closure of all data dependencies?
- How precisely can the impact of a new change to the system be measured?
- Does improving one model or signal degrade others?
- How quickly can new members of the team be brought up to speed?

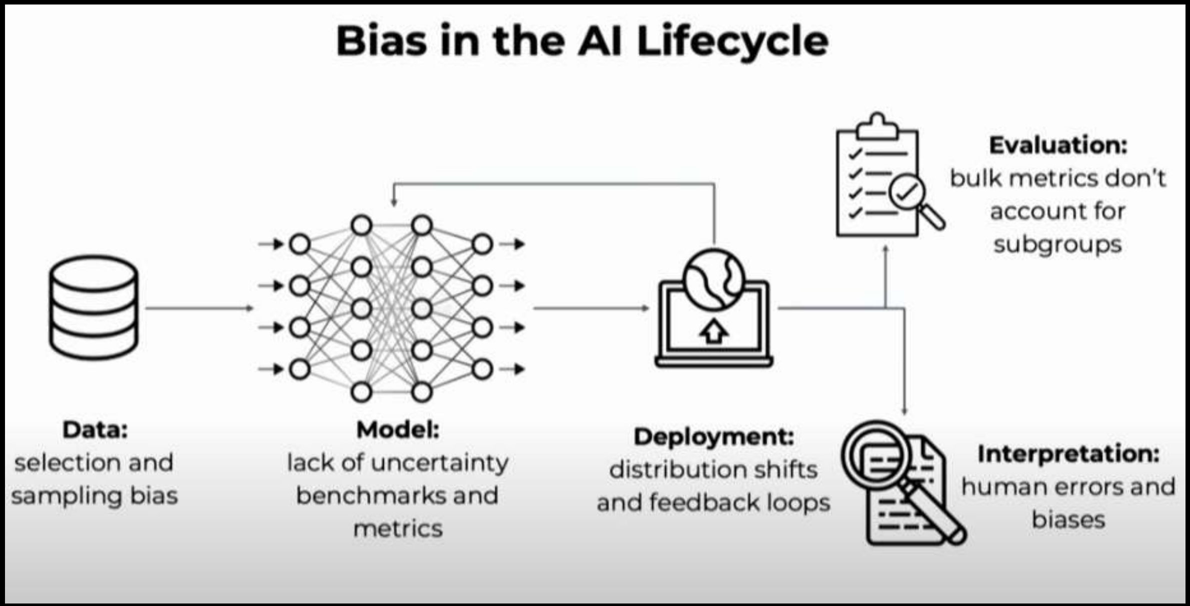
Robustness & Truswortheness

Barredo +, 2019





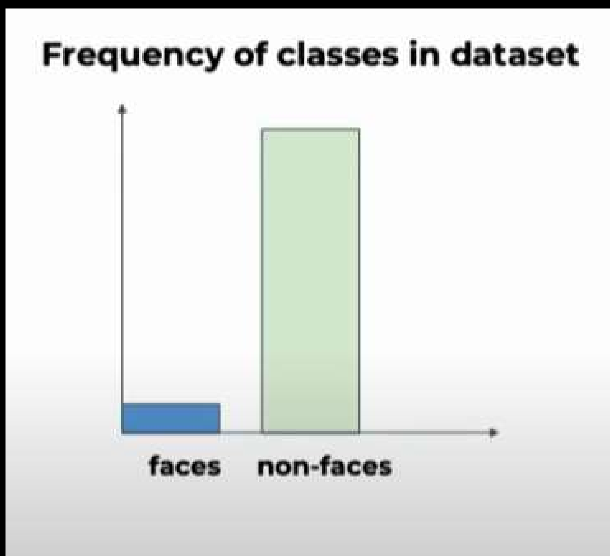
Robustness & Trusworthiness

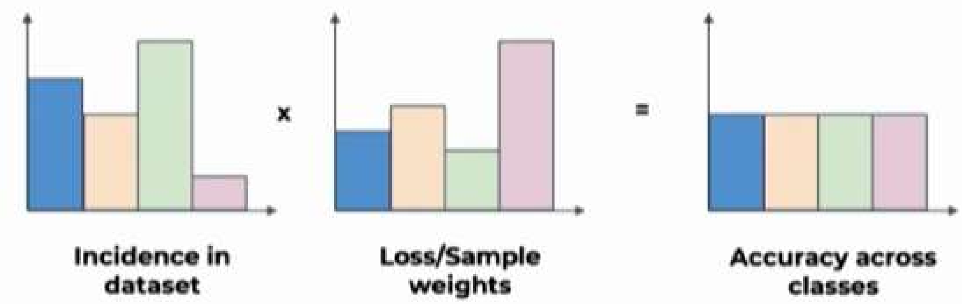


Robustness & Trusworthiness

Mitigating Class Imbalance

Class Imbalance

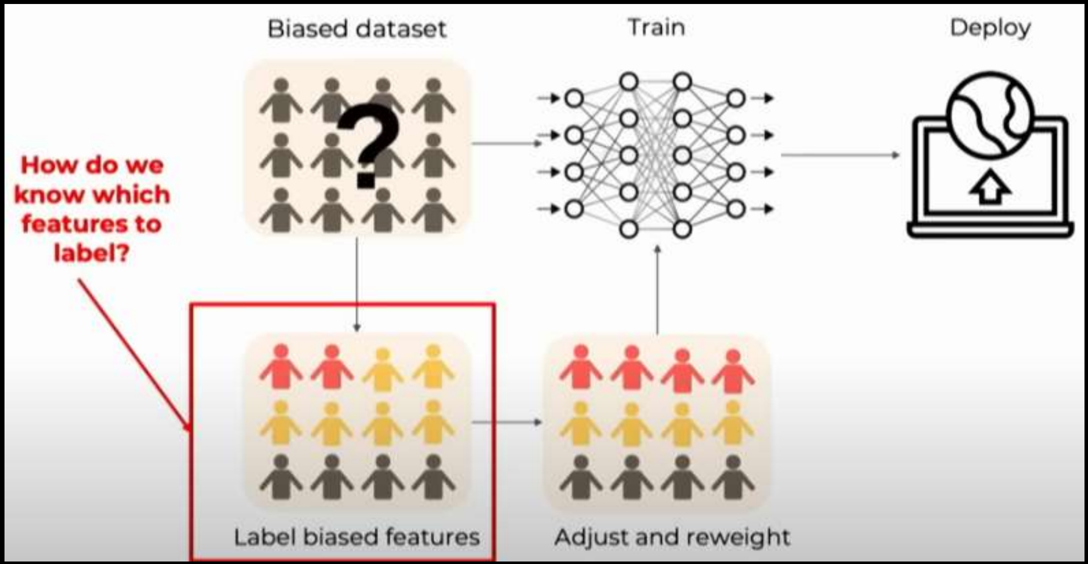




- **Sample Reweighting:** Sample more data points from underrepresented classes
- **Loss Reweighting:** Mistakes on underrepresented classes contribute more to loss
- **Batch Selection:** Choose randomly from classes so that every batch has an equal number of points per class

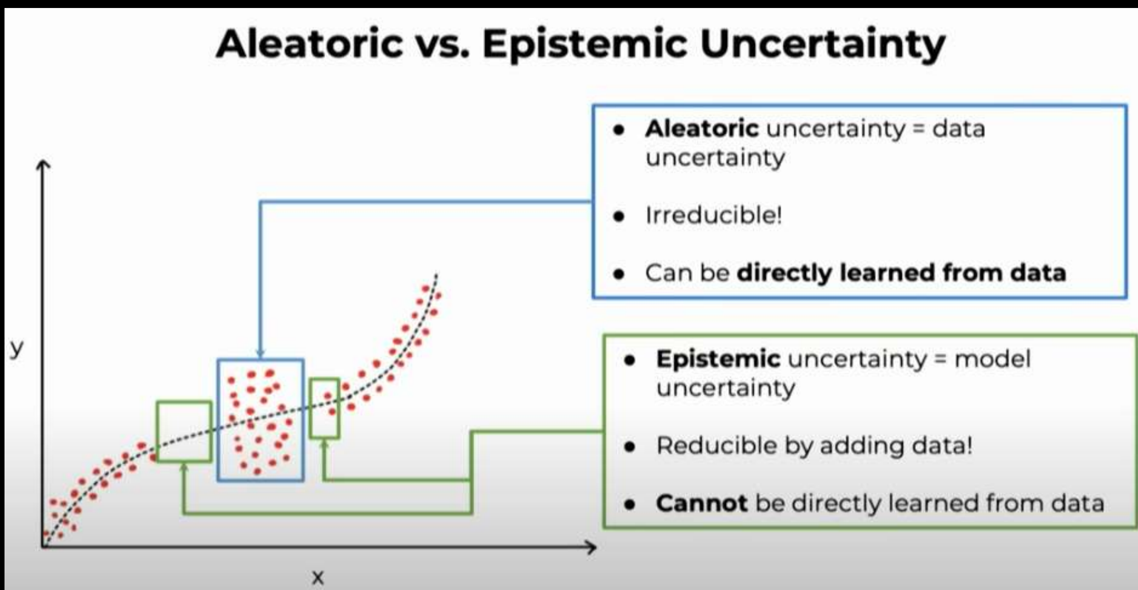
Robustness & Trusworthiness

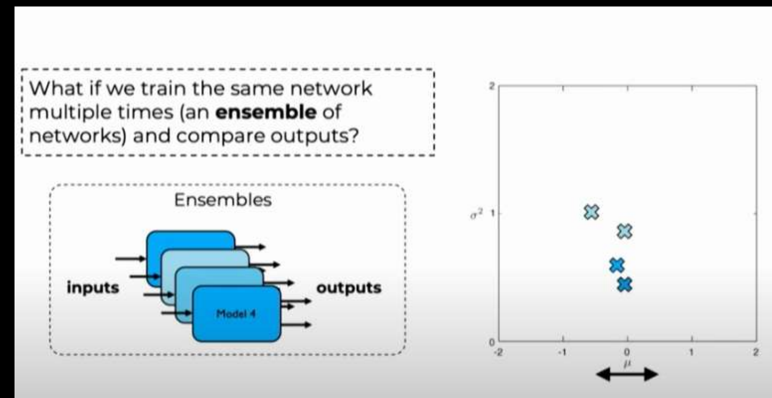
Latent features



Robustness & Trusworthiness

Uncertainty






```
num_ensembles = 5
for i in range(num_ensembles):
    model = create_model(...)
    model.fit(...)

raw_predictions = [models[i].predict(x)
    for i in range(num_ensembles)]
mu = np.mean(raw_predictions)
uncertainty = np.var(raw_predictions)
```

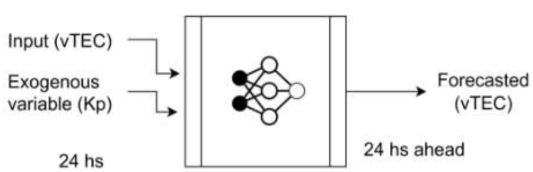
Molina +(submitted)

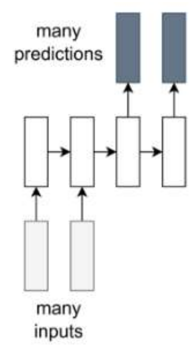
An application

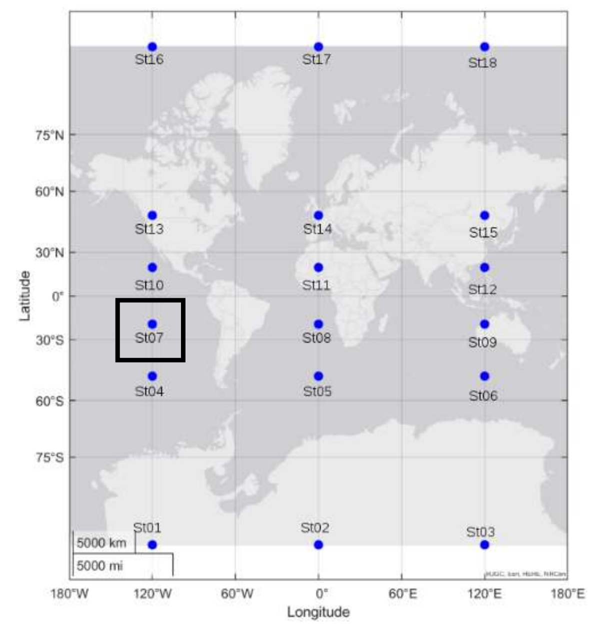
Objectives:

- Global TEC forecast 24 hs ahead using DL
- Propose a semi-operative prototype

- 2 stages: **a) single station forecasting (ML)**; b) extended forecasting
- 3 meridional sectors covering low, mid & high latitude
- Covering land & oceanic regions
- Input: TEC from GIMs + External input (Kp)







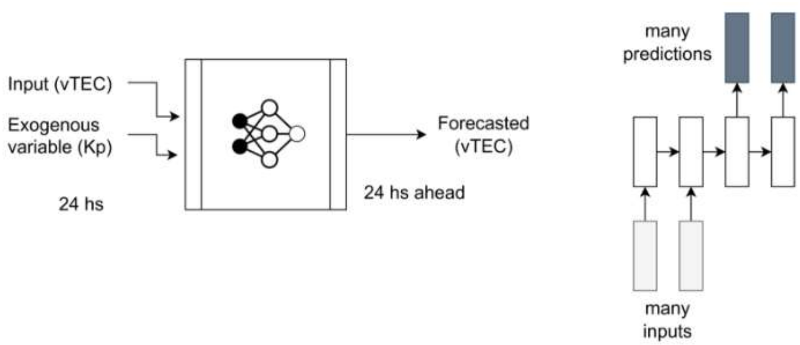
Cesaroni +, 2020

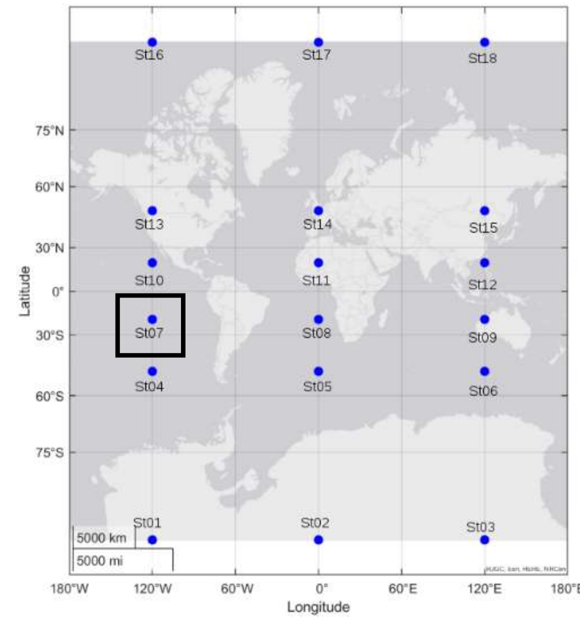
An application

Molina + (under review)

Objectives:
- Global TEC forecast 24 hs ahead using DL
- Propose a semi-operative prototype

- 2 stages: **a) single station forecasting (ML)**; b) extended forecasting
- 3 meridional sectors covering low, mid & high latitude
- Covering land & oceanic regions
- Input: TEC from GIMs + External input (Kp)





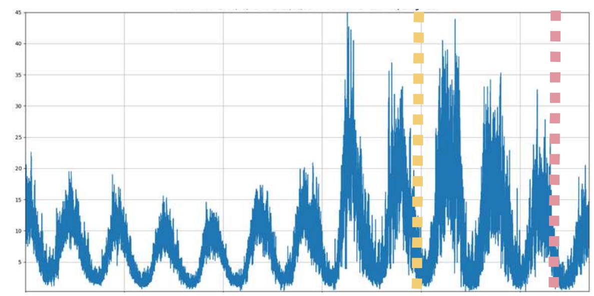
Cesaroni +, 2020

Data preparation & Feature selection

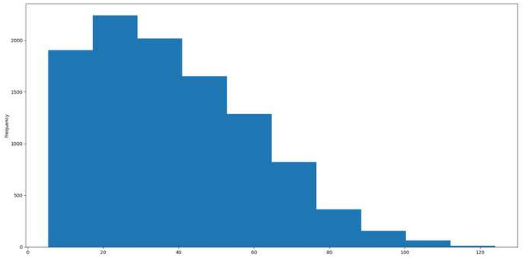
Loosely physics-informed approach

- Dataset:
  - 2005 - 2016
  - splitting strategy: 99% (99 train/1val) - 1% test (~43 days)
  - + cases study: geomagnetic storms in 2017
- Resolution (re-sampling):
  - TEC from GIMs - 2 hs resolution
  - Kp - 3hs resolution > K Nearest-neighbor interpolation
- Smart weight initialization (kernel initialization): GlorotNormal distribution + proper activation function (e.g. tanh).

St 01 TEC - dataset (2005 - 2016)

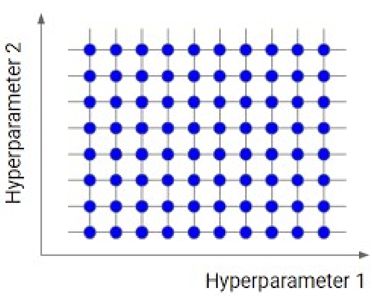


TEC - single ST Histogram

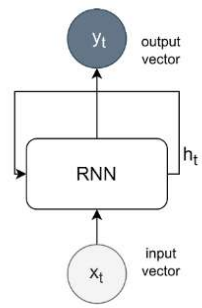


ML modelling

- 3 ML techniques:
  - 2 RNNs (LSTM & GRU)
  - CNN (1D)
- Time series
- Hyperparameter tuning: grid search



- # hidden layers (5,10,15,20,50,100 cells)
- batch size (16,32,64,128)
- #epochs (iterations) (5,10,15,20,30,40,50,100,200, 500,1000)

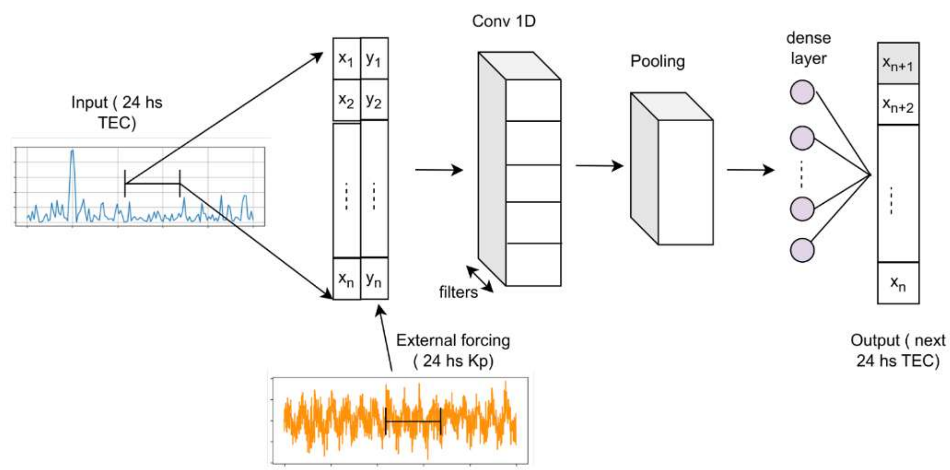


RNNs:

- Maintain order
- Memory (ht)
- Backpropagation through time
- Prone to overfitting, vanishing gradient problem
- LSTM & GRU -> gated cells -> long-term but not that long

CNN:

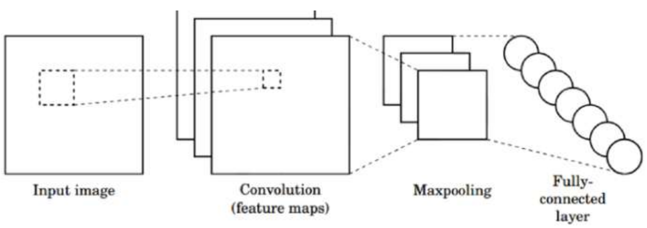
- kernel size = 2



ML modelling

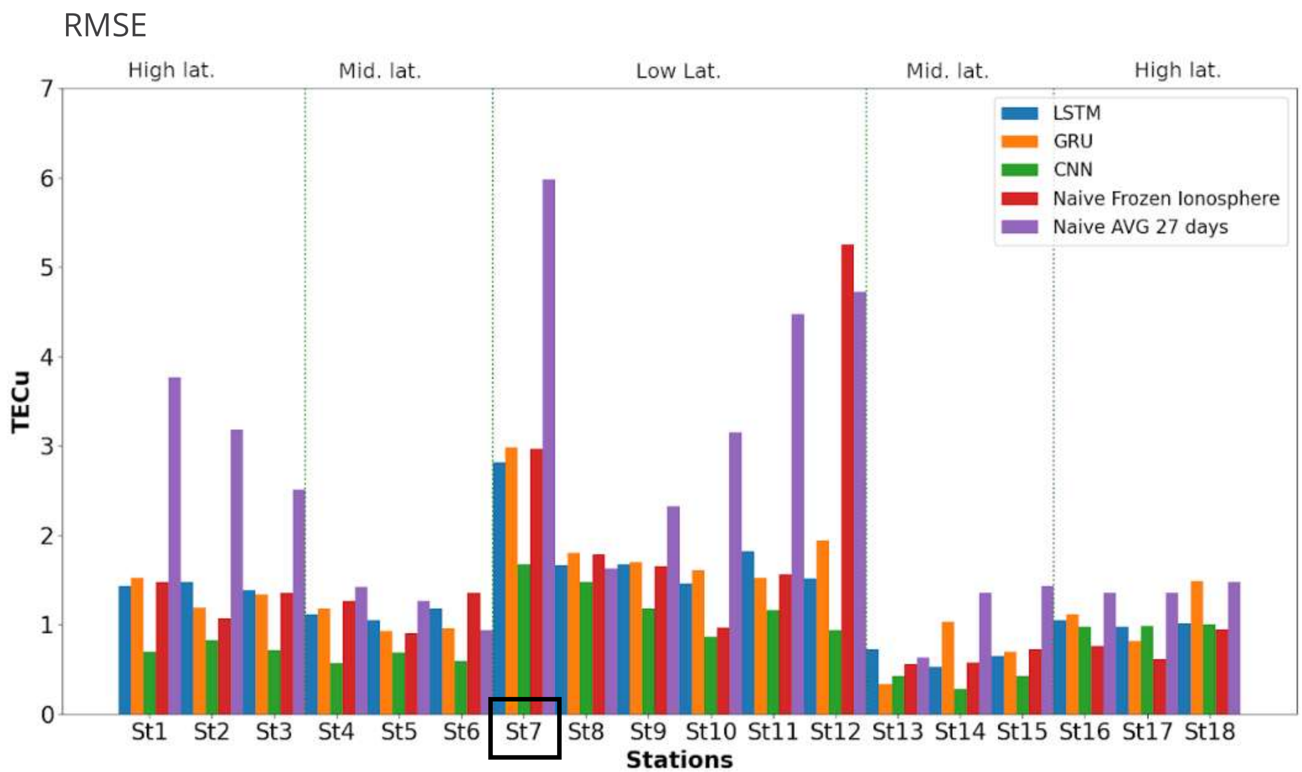
- Forecasting 24 hs ahead (quiet day)
- RMSE < 3 TECu
- CNN best at any station (– St16,17,18 -> TECu<=1 -> quiet day)
- Low lat + oceanic stations -> + challenging

- Why these results?
  - LSTM & GRU -> difficult to catch fast changes and peaks
  - CNN (1D) -> spatial relationship = short term relationships



TE	K
t0	t0
t1	t1
...	...

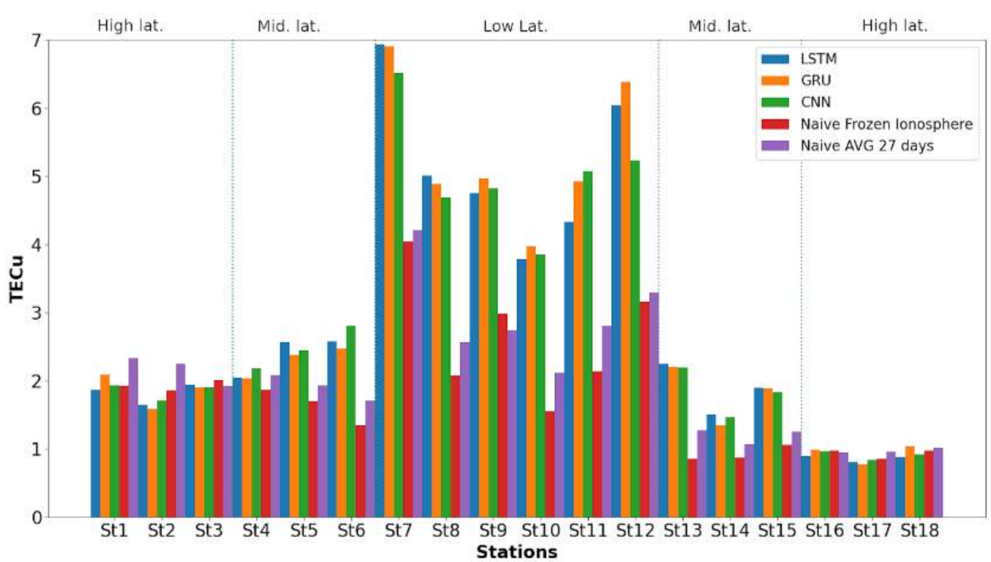
Kernel = 2



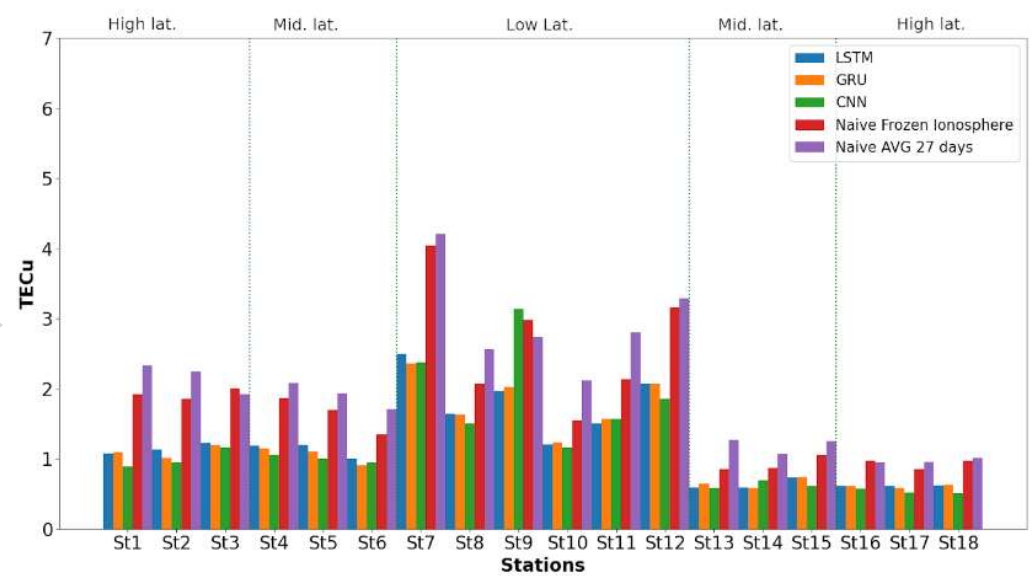
ML modelling

- In general: in SWx, few extreme cases (unbalanced datasets) -> forecasting may fail when new data arrives (generalization is a problem)-> Incremental learning

RMSE



Test set -> 43 days wiht the basic models

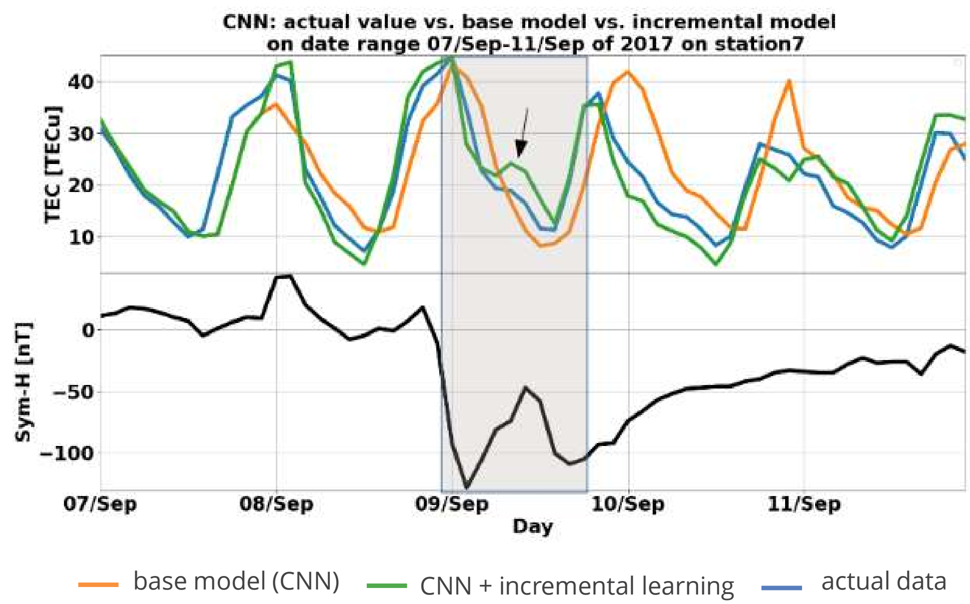


Test set -> 43 days with the models + incremental learning (updating each 24 hs)

ML modelling

$$Global\ \Delta TEC = \frac{1}{st} \sum^{st} \Delta TEC$$

- We considered cases study from 2017 under different geomagnetic conditions



Molina +(submitted)