* Spatiu de lucru: **Anaconda & Jupyter Notebook**
* Structura folder de lucru: \*insert image here\*
* TODO: Biblioteci + de explicat fiecare model.summary() in parte

1. **Citire**

* Folosesc fisierele text pentru citirea denumirilor imaginilor si a claselor din care acestea fac parte.
* Etape:
* 1.Citirea imaginilor de antrenare

Utilizez date din : fisierul train.txt si imaginile din folderul train

* 2.Citirea imaginilor de testare

Utilizez date din : fisierul test.txt si imaginile din folderul test

* 3.Citirea imaginilor de validare (testare acuratete)

Utilizez date din : fisierul validation.txt si imaginile din folderul validation

* Procesul de citire:

Retin clasele in ordinea in care apar in fisier sub forma unei liste(train\_labels/validation\_labels), respectiv fiecare imagine sub forma unei liste bidimensionale (dimensiunea de 32 X 32) intr-o lista separata (train\_images/test\_images/validation\_iamges). Cele doua liste vor fi transformate ulterior in numpy arrays.

Redimensionez lista cu imagini (functia expand\_dims[[1]](#footnote-1)) intr-un shape de valoare 4, a 4 a valoare fiind numarul 1 care va corespunde ulterior cu input shape-ul modelului. (1 pentru gray scale (0-255 valori pentru pixeli), 3 pentru RGB). Amestec esantionul de date pastrand relatia de aparteneta dintre imagini si clase (functia shuffle[[2]](#footnote-2)) pentru a avea un set de date de antrenare cat mai neomogen.

1. **Prelucrare date**

Tranform validation\_labels/train\_labels sub forma unei matrice binara (functia to\_categorical[[3]](#footnote-3)), pentru a putea utiliza in model categorical\_crossentropy ca loss function.

1. **Model CNN (Convolution Neural Network)**

* Modelul secvential (keras.Sequential):

API-ul secvențial permite crearea modelor strat cu strat (layer cu layer), dar este limitat prin faptul că nu permite crearea unor modele care partajează straturi(layers) sau au mai multe intrări sau ieșiri, precum cel functional. Deoarece pentru clasificarea imaginilor am ales un model mai simplist, utilizez API-ul de tip secvential( model= keras.Sequential).

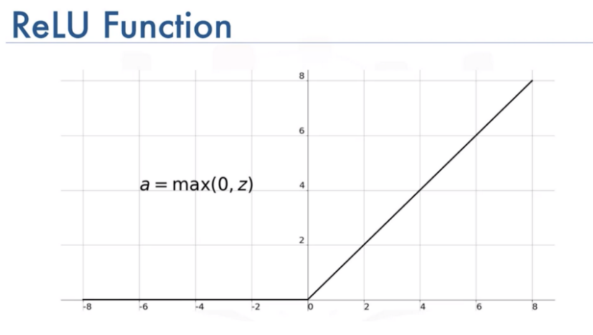
* Nucelul (kernel)

În procesarea imaginilor, nucleul este o matrice de convoluție care poate fi utilizata pentru estompare, ascuțire, reliefare, detectare a muchiilor și multe altele, făcând o convoluție între un nucleu și o imagine.

* Keras Conv2D

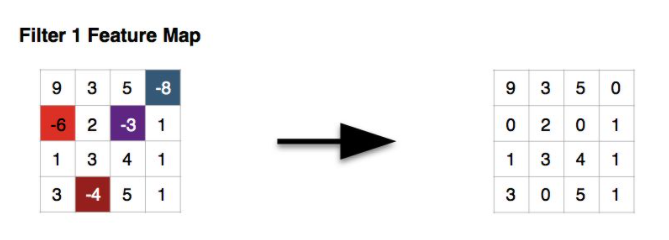
Este un strat(layer) de convoluție 2D care creează un nucleu(kernel) de convoluție care are intrări de straturi care ajută la producerea unui tensor de ieșiri.

**tf.keras.layers.Conv2D(filters, kernel\_size, activation)**

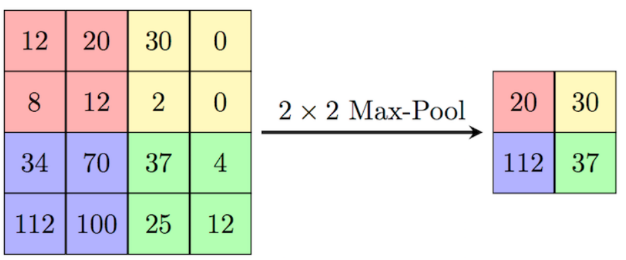
* filters: intreg care reprezinta dimensiunea spațiului de ieșire (adică numărul de filtre de ieșire în convoluție)
* kernel\_size: întreg(aceeași valoare pentru toate dimensiunile spațiale) /tuplu /listă de 2 numere întregi, specificând înălțimea și lățimea matricei de convoluție
* activation: functia de activare care va fi aplicata dupa convolutie. Dacă nu este specificat nimic, nu se aplică nicio activare.
* ReLU (Rectified Linear Unit)

-cel mai des utilizata in retele neuronale

-usor de antrenat

-performante bune

* Max Pooling2D

 Reduce datele de intrare luând valoarea maximă peste matricea cu dimensiunea pool\_size, prin deplasarea ferestrei pe toata matricea de intrare. Dimensiune output: output\_shape = (input\_shape - pool\_size + 1) / strides), strides=None (implicit)

**tf.keras.layers.MaxPooling2D(pool\_size=(2, 2) )**

* Flatten
* “Aplatizează” intrarea
* Este procesul de conversie a tuturor matricilor bidimensionale rezultante într-un singur vector liniar continuu
* Dintr-un input de (None, 4, 4, 64) => Un output de (None, 1024), 1024=4 \* 4\*64
* Dense
* Softmax
* Convertește un vector real într-un vector de probabilități categorice.
* Elementele vectorului de ieșire sunt în intervalul (0, 1) și însumează 1.
* Fiecare vector este tratat independent. Parametrul axis stabilește pe ce axă a inputului va fi funcția aplicată.
* Softmax este adesea folosit ca activare pentru ultimul strat al unei rețele de clasificare, deoarece rezultatul ar putea fi interpretat ca o distribuție de probabilitate.
* Softmaxul fiecărui vector x este calculat ca exp (x) / tf.reduce\_sum (exp (x))

- tf.math.exp(x): exponentialul lui x => y=ex

- tf.math.reduce\_sum(input\_tensor): suma elementelor unui tensor

* Parametrii:

x: Tensor de intrare.

axis: întreg, axă de-a lungul căreia se aplică normalizarea softmax.

* Returneaza:

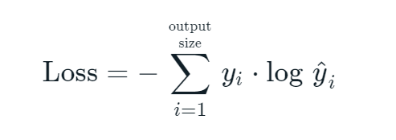
Tensor, ieșirea transformării softmax (toate valorile sunt non-negative și însumează 1)

**tf.keras.activations.softmax(x, axis=-1)**

1. **Configurare, antrenare, evaluare si prezicere**

* Compile(…), evaluate(…),fit(…) si predict(…) sunt metode a lui tf.keras.Model(…) - Modelează grupurile de straturi într-un obiect cu caracteristici de antrenament și inferență
* Configurare**:**

**model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])**

* + metrics : lista lucrurilor care trebuie evaluate de model în timpul antrenamentului și testării.
  + Categorical crossentropy:

- calculeaza valoarea loss al unui esantion dupa formula din dreapta

* + Optimizarea Adam:

- metodă care se bazează pe estimarea adaptivă a momentelor de ordinul întâi și de ordinul al doilea

*- Eficientă din punct de vedere al calculului, are puține cerințe de memorie, invariantă pentru redimensionarea diagonală a gradienților și este foarte potrivită pentru problemele mari în ceea ce privește datele / parametrii[[4]](#footnote-4)*

* Antrenare**:**

**model.fit(x=None, y=None, batch\_size=None, epochs=1, verbose=1,** **validation\_data=None)**

* + x: inputul (imaginile de training)
  + y: outputul (clasele imaginilor de training)
  + batch\_size: număr de eșantioane actualizate per gradient (default 32)
  + epochs: numărul de epoci pentru instruirea modelului.O epocă este o iterație asupra întregii date x și y furnizate
  + verbose=1 : progress bar afisat
  + validation\_data: tuplu format din imaginile de validare si clasele corespunzatoare
* Evaluare**:**

**model.evaluate(x,y)**

* returnează valoarea loss și valorile acuratetii(metrics) pentru model în modul test (imaginile de validare si clasele corspunzatoare)
* Prezicere**:**

**model.predict\_classes(x)**

* returneaza un array cu clasele corespunzatoare imaginilor de testare

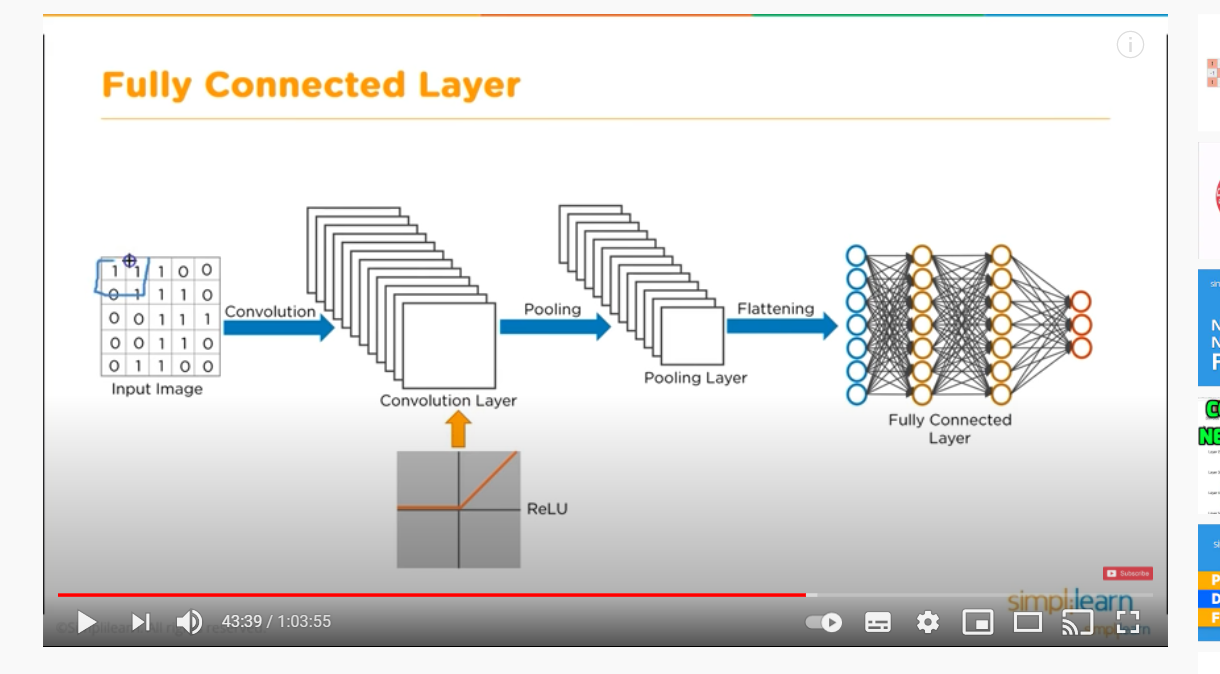
1. **Matrice de confuzie**
2. **Scriere in fiser CSV**

Input layer –accepta pixelii imaginii ca input sub forma de numpy array

Hidden layers – extragere feature prin calculare si manipulare

- Convolution layer, ReLU layer, Pooling layer, Fully Connected Layer – performeaza feature extraction din imagine

Output layer- layer complet conectat care identifica obiectul din imagine



1. numpy.expand\_dims (a, ax) - extindere forma (shape) array. Introducerea unei noi axe care va apărea în poziția axei în forma matricei extinse. Parametrii: a (numpy array/ list – array de modificat) si ax (intreg sau tuplu de intregi - poziția în axele extinse unde este plasată axa nouă (sau axele). Parametrul random\_state determină aleator generarea de numere pentru amestecarea datelor. [↑](#footnote-ref-1)
2. sklearn.utils.shuffle (\*arrays, random\_state = None, n\_samples = None) - amestecare arrays într-un mod consecvent. Acesta este un alias de conveniență pentru resample (\*arrays, replace=False) pentru a face permutări aleatorii ale colecțiilor. [↑](#footnote-ref-2)
3. tf.keras.utils.to\_categorical (y, num\_classes = None, dtype = 'float32')- Convertire vector întregi ([0, num\_classes]) în matrice de clase binare. [↑](#footnote-ref-3)
4. [Kingma et al., 2014](https://arxiv.org/abs/1412.6980) [↑](#footnote-ref-4)