Emotion Detector

Proiect realizat de: Morar Alexandru Flavius
Marunt Mirela Georgiana

Disciplina: Vedere artificiala pentru vehicule
An: 2022

Cuprins

Abstract

In zilele noastre, recunoașterea emoțiilor devine un subiect popular de deep-learning și oferă mai multe domenii de aplicare.

Ideea proiectului nostru este de a procesa imaginile de intrare (expresii faciale umane) care surprind diverse emoții cu scopul de a antrena modelele pre-antrenate pe seturi de date. Metodele convenționale deja existente necesită un extractor de caracteristici faciale, realizat manual de "facial Action Units" (AUs) pentru a extrage caracteristica specifica rezultata din "Facial Landmark regions", iar aceste fragmente action units extrase sunt procesate ulterior prin algoritmii tradiționali de machine-learning, cum ar fi Nearest Neighbours și SVM, care este un tip tipic de clasificator liniar.

Cu toate acestea, exista probleme in ceea ce priveste metoda convențională, si anume, sunt variatii ale nivelului de luminozitate, sau exista diverse pozitii din care obiectul este surprins (poate fi obturat) si toate aceste fapte pot conduce la deteriorarea vectorului in asa fel încât acuratetea este mult redusă. În plus, de cele mai multe ori, este dificila de realizat inginerie de caracteristici pentru a se potrivi cererii de recunoaștere facială.

Prin urmare, in cadrul acestui proiect, abordăm problema folosindu-ne de metoda de deep-learning a retelelor neuronale convulotionale(numite si "CNN"), care integrează pasul manual de extragere a feature-ului specific unei expresii faciale cu un clasificator de training. Acest mod de lucru este capabil să atinga soluția relativ cea mai optimă prin procesul de retropropagare în care algoritmul învață ponderile prin coborârea gradientului stocastic modificat care poate găsi direcțiile care minimizează cel mai bine pierderea de la rezultatul de baza. Astfel, rezultatul numeric al algoritmului va arăta un rezultat probabilistic a fiecărei clase etichetate.

In concluzie, pentru a reduce costurile de calcul, se aplică tehnica de reglare "fine-tuning" astfel încât putem adapta un model pre-antrenat la variația setului nostru de date locale. Ca rezultat, o astfel de metodă este cea mai eficienta si optima pentru rezolvarea problemelor ce pot aparea in urma variațiilor de iluminare și a orientării diferite a obiectului din imagine cu scopul de a obține un nivel de precizie cat mai ridicat.

Introducere

Expresiile faciale sunt unul dintre principalele subiecte abordate in recunoasterea faciala atunci cand ne referim la emoţiile umane, care poate genera aplicatii atat in directii tehnice cat si aplicabile in viata de zi cu zi, putand fi utilizate în afara experimentelor de laborator.

Această proiecţie consta in implementarea unui un model bazat pe deep-learning pentru a clasifica o anumită expresie faciala dintr-o imagine într-una dintre cele şapte emoţii umane de bază (fericire, furie, tristete, neutru, surprindere, dezgust si frica).

Modul în care construim acest model este prin realizarea transferului de invatare a unui model preantrenat existent, iar rezultatele testarii vor fi comparate si validate pe baza acurateţii modelului. Sarcinile acestui proiect includ preprocesarea datelor (adica a imaginilor), augmentarea setului de date restrans detinut, testarea inaintea antrenarii modelului, antrenarea propriu-zisa a modelului, iar in cele din urma predicţia si evaluarea.

O prezentare vizuală a rezultatului va fi similară cu următoarea figura(Fig1).

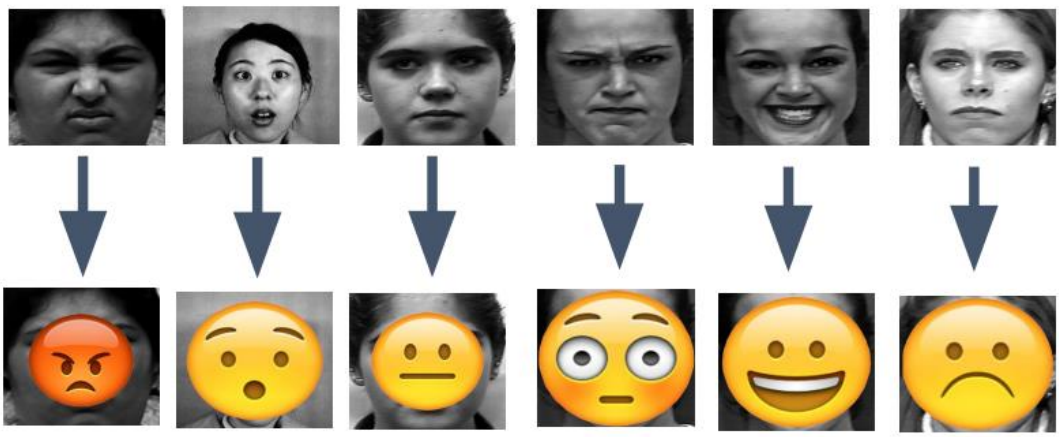


Fig1. Simularea asignarii expresiilor faciale cu emotiile corespunzatoare(folosindu-ne de emoji)

Tehnologii si biblioteci utilizate

In cadrul realizarii acestui proiect ne-am folosit de urmatoarele tehnologii si biblioteci:

- Workspace: Anaconda, Jupyter Notebook, Visual Studio Code
- Tensorflow, Keras (layers, models, preprocessing.image)
- Matplotlib
- Numpy
- Pandas
- Seaborn
- Dataset - Kaggel

Pasi creare si realizare proiect:

Instalare Anaconda - > Creare folder nou de proiect - > Deschiderea unui comand prompt de Anaconda - > Crearea unui environment si instalarea tensorflow (activarea acestuia - > "conda activate tf") -> Deschiderea unui jupyter notebook (comanda in command prompt: "jupyter notebook") - > Crearea unui fisier python - > Importarea si instalarea de bibiloteci necesare in vederea realizarii proiectului - > Importarea setului de date si afisarea acestora - > Antrenarea si validarea setului de date (ne folosim de folderele de train si validation din setul de date) - > Crearea modelului si exportarea acestuia - > Integrarea dintre modelul creat cu datele de integrare si validare (ne folosim de partea de deep-learning - > keras) - > Deschidere Visual Studio Code - > creare fiser main - > implementare - > rulare

Despre tehnologii si biblioteci:

- *Anaconda*

Anaconda este o distribuție a limbajelor de programare Python și R pentru calculul științific (data science, aplicații machine learning, procesarea datelor la scară largă, analiză predictivă), care își propune să simplifice gestionarea și implementarea pachetelor. Distribuția include data science packages potrivite pentru Windows, Linux și macOS.

Distribuția Anaconda vine cu peste 250 de pachete instalate automat și peste 7.500 de pachete open-source suplimentare pot fi instalate din PyPI, precum și pachetul conda și managerul de mediu virtual. De asemenea, include o interfață grafică, Anaconda Navigator, ca alternativă grafică la interfața de linie de comandă (CLI).

Marea diferență dintre conda și managerul de pachete pip constă în modul în care sunt gestionate dependențele pachetelor, ceea ce

reprezinta o provocare semnificativă pentru data science realizat in Python și motivul pentru care conda există.
Conda este un sistem de management al environment-ului si manager de pachete, de tip open-source, cross-platform, language-agnostic care instalează, rulează și actualizează pachetele și dependențele acestora. A fost creat pentru Python, dar poate împacheta și distribui software pentru orice alt limbaj (de exemplu, R), inclusiv proiecte în mai multe limbaje. Pachetul conda și managerul de environment sunt incluse în toate versiunile Anaconda, Miniconda și Anaconda Repository.

- *Jupyter Notebook*

Jupyter Notebook (fost IPython Notebooks) este un web-based environement computational interactiv pentru crearea de documente de tip notebook.

Un document Jupyter Notebook este un REPL bazat pe browser, care conține o listă ordonată de celule de intrare/ieșire care poate conține cod, text (folosind Markdown), expresii matematice, diagrame și conținut "rich media". In esenta, un notebook este un document JSON, urmând o schemă de versionare, care se termină de obicei cu extensia „.ipynb".

Notebook-urile Jupyter sunt construite pe o serie de biblioteci open-source populare:

- ❖ IPython
- ❖ ZeroMQ
- ❖ Tornadă
- ❖ jQuery
- ❖ Bootstrap (cadru front-end)
- ❖ MathJax

Jupyter Notebook se poate conecta la mai multe kernel-uri pentru a permite programarea în diferite limbaje. Un nucleu Jupyter este un program responsabil cu gestionarea diferitelor tipuri de solicitări (execuția codului, completările codului, inspecția) și furnizarea unui răspuns. Kernel-urile vorbesc cu celelalte componente ale Jupyter folosind ZeroMQ și, astfel, pot fi pe aceleași mașini sau remote. Spre deosebire de multe alte interfețe asemănătoare notebook-urilor, în Jupyter, nucleele nu știu că sunt atașate la un anumit document și pot fi conectate la mai mulți clienți simultan. De obicei, nucleele permit executarea unui singur limbaj, dar există câteva excepții.

Un Jupyter Notebook poate fi convertit într-un număr de formate de ieșire standard deschise (HTML, slide-uri de prezentare, LaTeX, PDF, ReStructuredText, Markdown, Python) prin „Download as” în interfața web, prin biblioteca nbconvert sau „jupyter nbconvert" in interfața de linie de comandă dintr-un shell.

- *Visual Studio Code*

Visual Studio Code, denumit și VS Code, este un editor de cod sursă realizat de Microsoft pentru Windows, Linux și macOS. Caracteristicile includ suport pentru depanare, evidențierea sintaxei, completarea inteligentă a codului, fragmente, refactorizarea codului și Git încorporat.

Suporta limbaje de programare precum: Java, JavaScript, Go, Node.js, Python, C++ si Fortran.

Spre deosebire de un project system, acesta permite utilizatorilor să deschidă unul sau mai multe directoare, care pot fi apoi salvate în spații de lucru pentru reutilizare ulterioară. Acest lucru îi permite să funcționeze ca un editor de cod agnostic pentru orice limbaj. Acceptă multe limbaje de programare și un set de caracteristici care diferă în funcție de acest lucru. Fișierele și folderele nedorite pot fi excluse din arborele proiectului prin intermediul setărilor. Multe caracteristici Visual Studio Code nu sunt expuse prin meniu sau prin interfața cu utilizatorul, dar pot fi accesate prin linia de comenzi.

- *TensorFlow*

TensorFlow este o bibliotecă de software gratuită și open-source pentru machine-learning și inteligența artificială. Poate fi folosit într-o gamă largă de sarcini, dar se concentrează în special pe training și interpretarea rețelelor de tip deep-neural.

- *Keras*

Keras este o bibliotecă de software open-source care oferă o interfață Python pentru rețelele neuronale artificiale. Keras acționează ca o interfață pentru biblioteca TensorFlow.
Keras conține numeroase implementări ale builder block-urilor rețelei neuronale, cum ar fi layers, obiective, funcții de activare, optimizatori și o serie de instrumente pentru a simplifica lucrul cu

date (imagine și text) pentru a simplifica implementarea rețelei deep-neural.

- *Matplotlib*

Matplotlib este o bibliotecă de plotare pentru limbajul de programare Python și extensia sa numerică de matematică NumPy. Oferă un API orientat obiect pentru încorporarea de plot-uri în aplicații folosind seturi de instrumente GUI de uz general precum Tkinter, wxPython, Qt sau GTK.

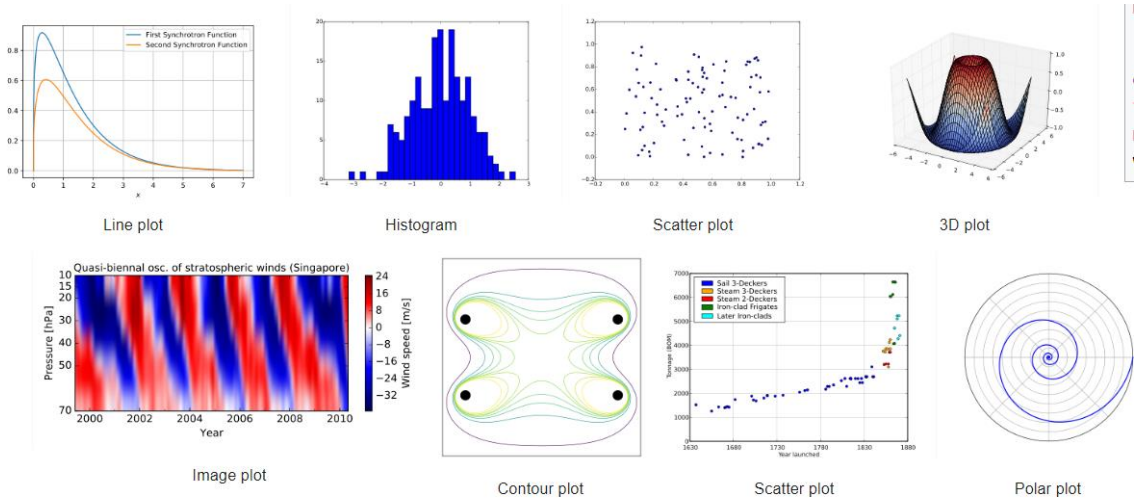


Fig 2. Exemple de matplot

- *Numpy*

NumPy este o bibliotecă pentru limbajul de programare Python, adăugând suport pentru matrice și matrice mari, multidimensionale, împreună cu o colecție mare de funcții matematice de nivel înalt pentru a opera pe aceste matrice.

- *Pandas*

Pandas este un pachet Python open-source care este utilizat pe scară largă pentru data science/data analysis și task-uri de machine learning. Este construit ca extindere a pachetului Numpy, care oferă suport pentru matrice multidimensionale.

- *Seaborn*

Seaborn este o bibliotecă care folosește Matplotlib pentru a reprezenta grafice. Va fi folosit pentru a vizualiza random distributions.

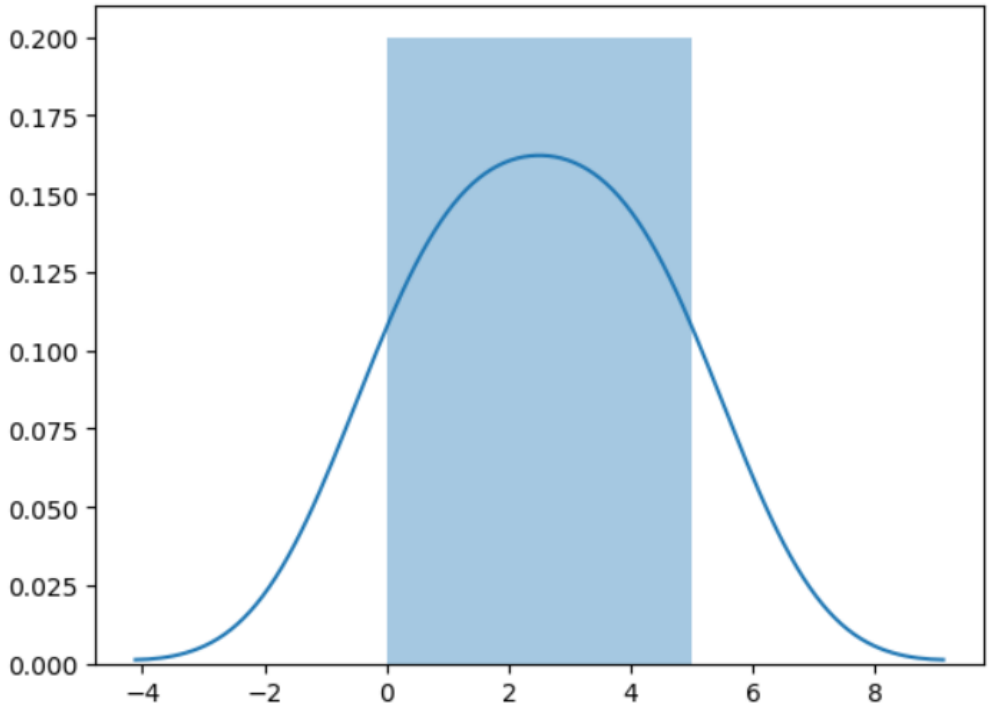


Fig 3. Exemplu de vizualizare grafic folosind Seaborn

Metode, concepte si algoritm

Ca toate celelalte probleme de clasificare, problema recunoașterii emoțiilor necesită un algoritm pentru extragerea completă a trăsăturilor și clasificarea pe categorii. Pentru a clasifica o emoție, trebuie sa extragem anumite feature-uri din datele pe care le avem si sa construim un model care poate clasifica orice data (imagine) de input pe baza feature-urilor.

Procedura poate fi subliniată după cum urmează:

1. Pre-procesare a datelor:

Pre-procesarea datelor se foloseste pentru a standardiza datele. Modul tipic de a realiza acest lucru este prin a seta media datelor la 0 și, de asemenea, divizarea datelor la abaterea standard.

2. Extragerea feature-urilor:

Metoda convențională tipică este detectarea feței și extragerea unităților de acțiune (AU) (prezentate în figura 4) de pe expresia faciala, cautand combinația de cod "AU" pentru a identifica anumite emoții.

3. Construcție model:

Clasificatorul convențional poate fi fie un algoritm supravegheat, fie nesupravegheat. Un tipic exemplu de algoritm supravegheat este Support Vector Machine, iar exemplele de algoritm nesupravegheat include Principle Component Analysis(PCA) and Linear Discriminant Analysis (LDA).

4. Generarea categoriilor/rezultatelor:

Modul tipic de a genera o categorie sau un rezultat este de a găsi limita de decizie care are minimul de distanța euclidiană față de date.

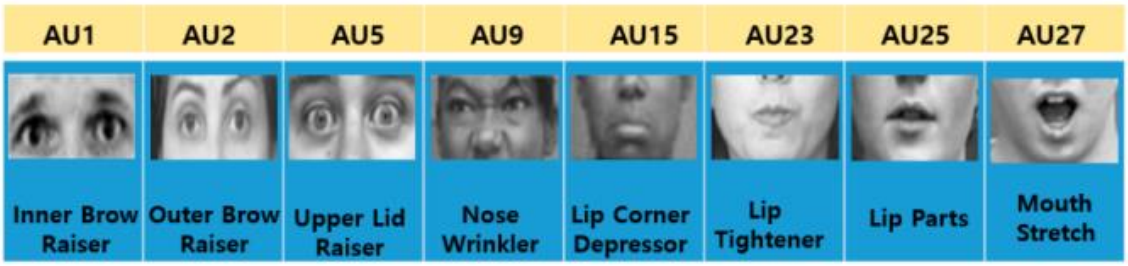


Fig 4. Exemple de Action Units (AU)

Dupa cum precizam si anterior, exista 2 tipuri de probleme care apar in cadrul metodelor conventionale:

- Variatia nivelului de luminozitate

Avand in vedere faptul ca fiecare imagine este realizata într-un backgroud diferit și condiții de iluminare complet diferite, intra-clasa de light-noise va distorsiona modelul pentru a putea clasifica emoția. Astfel, ca rezultate, unele tipurile de emoții pot fi clasificate complet diferit din cauza efectului de lighting-noise.

- Variatia de locație

Deoarece feature-ul este de obicei extras de filtre, astfel de aplicații ale modelului binar local în locația feature-ului, poate afecta funcționalitatea de extragere a caracteristicii. Ca rezultat, AU poate fi extras incorect dacă fața este rotită sau este într-o parte diferită a imaginii, ori daca este obturata.

În acest proiect, pentru a rezolva problemele existente cu metodele conventionale, ne folosim de o tehnică de deep learning numită rețele neuronale convoluționale (CNN) pentru a genera un model de clasificare care combină extragerea caracteristicilor cu clasificarea.

Retelele neuronale convolutionale (CNN) se diferentiaza de cele conventionale prin cateva specificatii cheie:

- Generarea automata de feature de extragere:

Caracteristica imaginilor poate fi capturată automat fără feature-ul de extragere construit de utilizatori deoarece extractoarele de caracteristici sunt generate automat în procesul de training pe baza datelor de input primite.

- Diferentele modelului matematic:

Metoda convențională realizează de obicei clasificarea prin transformare liniară, deci sunt denumite, de obicei, drept clasificatoare liniare. În comparație, CNN, precum și alti algoritmi de deep-learning combină de obicei transformarea liniară cu funcția neliniară, un exemplu ar fi sigmoid-ul (funcție logistică) și Rectified Linear Unit (ReLU) pentru a distinge diferențele în procesul de clasificare.

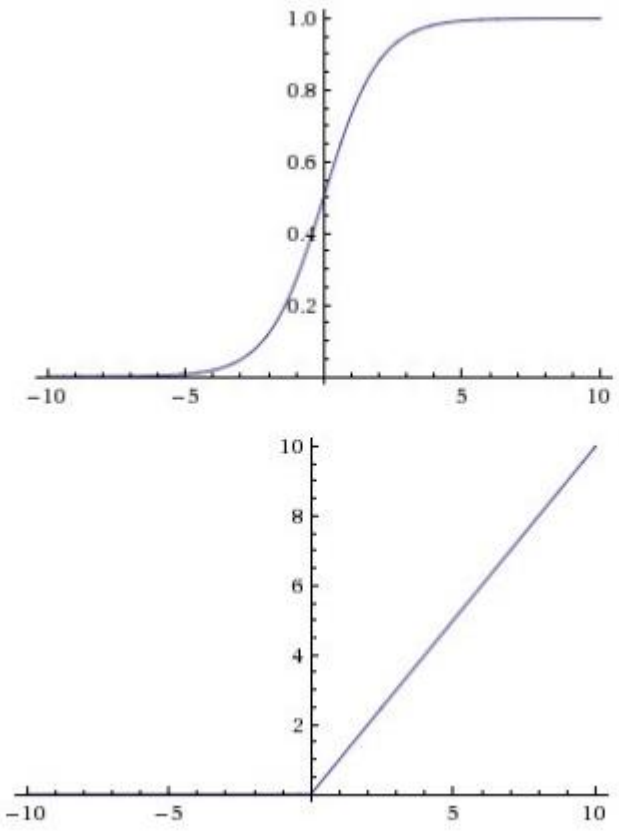


Fig 5. Functia sigmoid (primul grafic) si functia ReLU (graficul de jos)

- The Deeper Structure:

Metoda convențională realizează de obicei doar un singur nivel de operație: de exemplu SVM are un singur set de weights. Cu toate acestea, CNN, precum și alte tipuri de algoritmi deep-learning, realizeaza mai multe straturi de operare în procesul de clasificare.

Etape de implementare al algoritmului CNN:

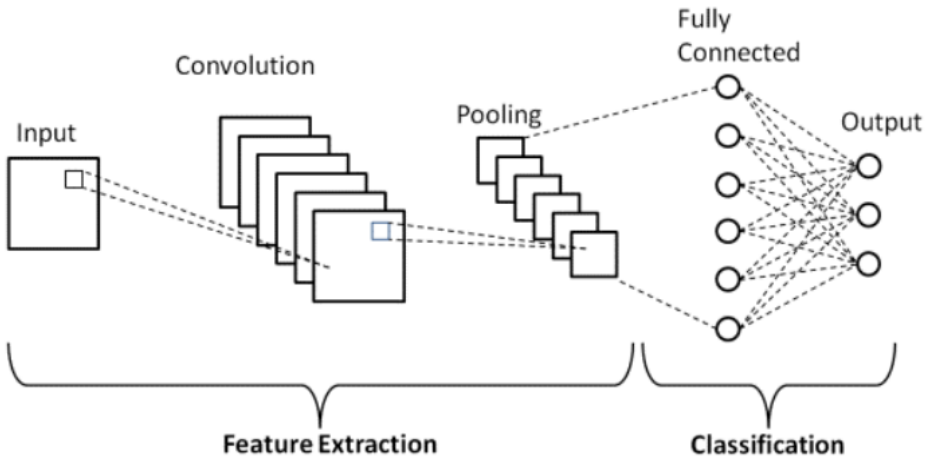
1. Pregătirea setului de date și importarea librăriilor

În acest moment dispunem de principala resursa a proiectului, setul de date în care sunt stocate imagini și care este ierarhizat după urmatoarea structura:

- date de antrenament (80% din date)
  - clasa 1 de imagini - angry
  - clasa 2 de imagini - disgust
  - clasa 3 de imagini - fear
  - clasa 4 de imagini - happy

    - clasa 5 de imagini - neutral
    - clasa 6 de imagini - sad
    - clasa 7 de imagini - surprise
- date de validare (20% din date)

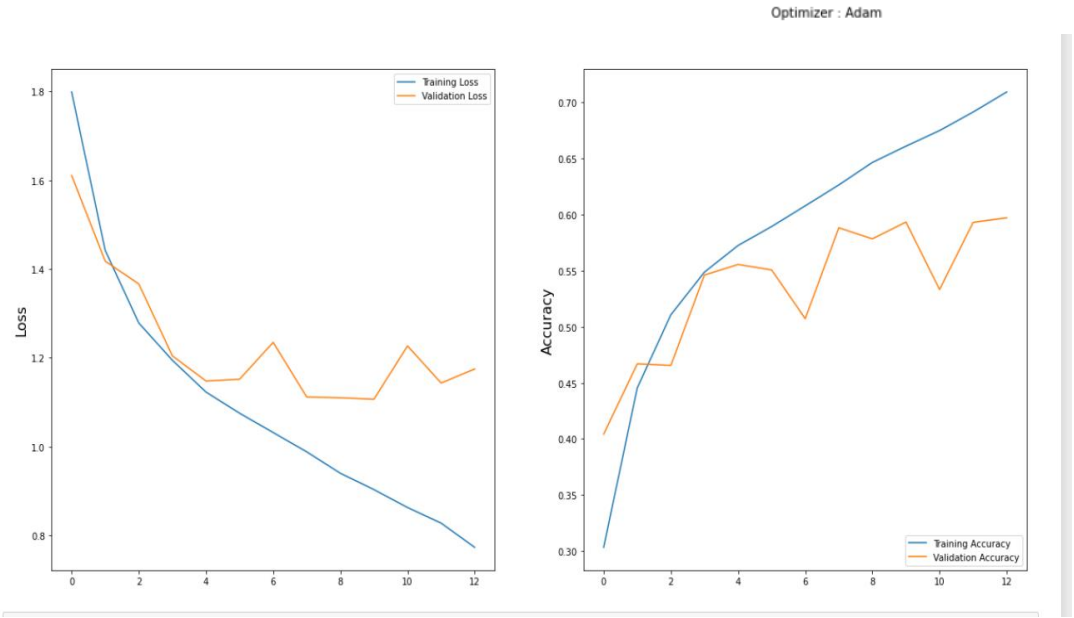
2. Vizualizarea si importarea imaginilor din setul de date
3. Crearea datelor de antrenament şi validare:
    a. Setarea unui batchsize: hiperparametru care ia din datele de antrenament un număr de imagini definit şi le antrenează în reţeaua neuronală
    b. Augmentarea datelor
4. Construirea Modelului CNN:
    a. Construirea unui model secvenţial permite formarea de straturi pas cu pas



5. Ajustarea modelului cu ajutorul unor parametri:
    a. ModelCheckpoint - este folosit împreună cu antrenamentul folosind funcţia model.fit() pentru a salva modelul (într-un fișier punct de control) la un anumit interval, astfel încât modelul sau greutățile pot fi încărcate ulterior pentru a continua antrenamentul din starea salvată. În cadrul proiectului se va crea un fişier cu extensia ".h5" (un fel de fișier de bibliotecă folosit pentru stocarea unor cantităţi mari de date numerice, grafice și text.)
    b. EarlyStopping - Opreşte procesul de antrenament atunci când o valoare monitorizată nu se mai îmbunătăţeşte.
    c. ReduceLROnPlateau - pe măsură ce modelul se ajustează în timpul compilării, iar valorile nu se îmbunătăţesc se va reduce rata de învăţare
    d. Compilare

6. Vizualizare rezultate:

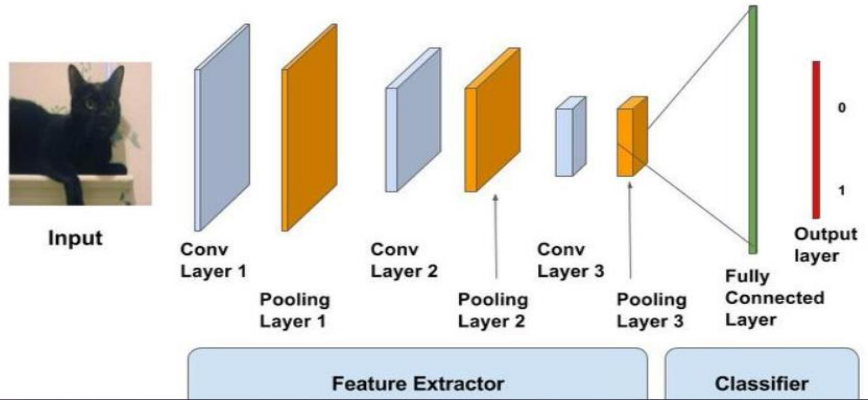
In [9]:
```
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



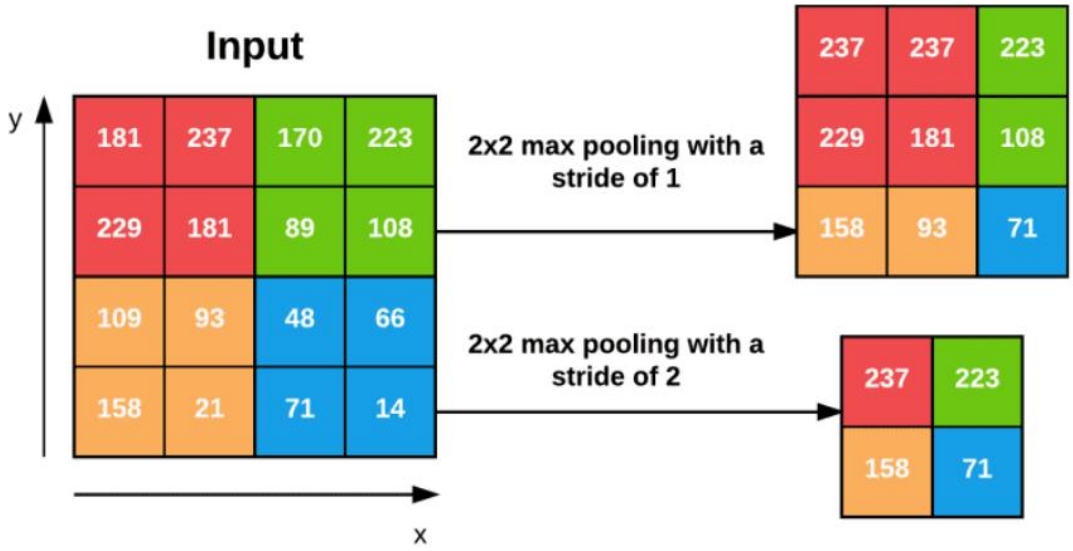
7.Structura Model CNN

A typical CNN model looks like this:

- Input layer
- Convolution layer + Activation function
- Pooling layer
- Fully Connected Layer



8.Max pooling



Directii viitoare

In prezenta lucrare am evidentiat aspecte referitoare la realizarea proiectului. De la tehnologii si biblioteci utilizate, la concepte si algoritmi utilizati, dataset alcatuit din imagini atat pentru training cat si pentru etapa de validare. De asemenea am precizat si pasii pe care i-am urmat in crearea si implemntarea proiectului.

Avand in vedere importanta si versatilitatea topic-ului lucrarii alese, propunem mai multe viitoare directii de dezvoltare, precum:

- In cadrul unei aplicatii pentru soferi (atunci cand acestia sunt obositi, prea nervosi, sau au o traire intensa care nu este adecvata in trafic) acestia sa primeasca notificari si alerte.

- In cadrul unei aplicatii pentru adolescenti, care in perioada adolescentei sunt predispusi la stari de anxietate, stres, atacuri de panica, depresie.

- In cadrul unei aplicatii care sa ne ajute sa ne dezvoltam skill-urile non-verbale.

Bibliografie

[1]https://www.upgrad.com/blog/basic-cnn-architecture/
[2]https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
[3]https://dspace.library.uvic.ca/bitstream/handle/1828/13388/Adil_Mohammed_Adnan_MEng_2021.pdf?sequence=3
[4]https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
[5]https://en.wikipedia.org/wiki/TensorFlow
[6]https://en.wikipedia.org/wiki/Visual_Studio_Code
[7]https://en.wikipedia.org/wiki/Matplotlib
[8]https://en.wikipedia.org/wiki/NumPy
[9]https://en.wikipedia.org/wiki/Keras
[10]https://en.wikipedia.org/wiki/Pandas_(software)
[11]https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook
[12]https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp
[13]https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/
[14]https://studymachinelearning.com/data-augmentation-2/
[15]https://studymachinelearning.com/keras-imagedatagenerator-with-flow_from_directory/
[16]https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
[17]https://www.upgrad.com/blog/basic-cnn-architecture/
[18]https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08