UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENŢĂ:
Informatică Aplicată

Recunoasterea semnelor de circulatie

COORDONATOR:
Conf. Dr. Cosmin BONCHIȘ

STUDENTI:
Chera Denis Marian
Constantinescu Marius Alexandru

TIMIŞOARA
2022

Cuprins

Capitolul 1

Descrierea proiectului

Scopul esential al acestui proiect este utilizarea tehnicilor de procesare a imaginilor pentru a ajuta in recunoasterea semnelor de circulatie distincte. Acest lucru va contribui la simplificarea vietii oamenilor si la imbunatatirea calitatii acesteia, reducand considerabil dependenta lor de ceilalti si, prin urmare, asigurandu-le siguranta in timpul conducerii. Abordarea actuala are mai multe defecte si preia mai putine caracteristici din fotografii. Metoda noastra sugerata utilizeaza o abordare a retelei neuronale convolutionale (CNN), care imbunatateste acuratetea recunoasterii si permite extragerea de informatii suplimentare.

Detectarea semnelor de circulatie este instruita in aceasta lucrare folosind o abordare de baza a procesarii imaginilor, care reduce timpul de procesare si imbunatateste acuratetea.

Capitolul 2

Prezentare generala

Este bine cunoscut faptul ca numarul de vehicule rutiere a crescut enorm datorita realizarilor tehnologice din industria auto si, foarte precis, disponibilitatii tarifelor mici. Odata cu aceasta crestere remarcabila, numarul accidentelor este de asemenea intr-o crestere accentuata de la an la an, din cauze diferite, in care necunoasterea semnelor de circulatie este considerata drept o cauza majora a acestor accidente.

Dezvoltarea sistemelor autonome de recunoastere a semnelor de circulatie ajuta la asistarea soferului in diferite moduri pentru a garanta siguranta acestuia, ceea ce pastreaza, de asemenea, si siguranta celorlalti soferi si pietoni.

Semnele de circulatie sunt utilizate ca metoda de avertizare si indrumare a soferilor, ajutand la reglarea fluxului de trafic printre vehicule, pietoni, motociclete, biciclete si altii care calatoresc pe strazi, autostrazi si alte drumuri. Semnele de circulatie sunt unul dintre echipamentele rutiere, sub forma de simboluri, litere, numere, propozitii si combinatii precum avertismente, interdictii, ordine sau indicatii pentru utilizatorii drumurilor. In incercarea de a se concentra asupra drumului in timpul conducerii, soferii rateaza adesea indicatoarele aflate pe marginea drumului, lucru care ar putea fi periculos pentru ei si pentru oamenii din jurul lor.

Capitolul 3

Tehnologii si librari utilizate

3.1 Limbajul de programare Python

Python este un limbaj de programare interpretat, interactiv, orientat pe obiecte. Acesta încorporează module, excepții, tastare dinamică, tipuri de date dinamice la nivel foarte înalt și clase. Pe langă aceste lucruri, oferă suport și pentru alte paradigme de programare cum ar fi cea procedurală și funcțională, nu numai pentru programarea orientată pe obiecte. Python are capacitatea de a combina o putere impresionantă cu o sintaxă extrem de clară.

De asemenea, are interfețe pentru mai multe apeluri de sistem și biblioteci, precum și pentru diferite sisteme de ferestre, fiind extensibil în C sau C++. De altfel, poate fi folosit ca limbaj de extensie pentru aplicațiile care necesită o interfață programabilă. De menționat este și faptul că Python este un limbaj de programare portabil, deoarece poate rula pe mai multe variante Unix, inclusiv Linux si macOS, cât și pe Windows. [3]

3.2 Librariile utilizate

3.2.1 Tensorflow

TensorFlow este o bibliotecă Python pentru calcul numeric rapid creată și lansată de Google. Este o bibliotecă de bază care poate fi folosită pentru a crea modele de deep learning direct sau prin utilizarea bibliotecilor wrapper care simplifică procesul construit cu TensorFlow.

Figura 3.1: Tensorflow

3.2.2 Keras

Keras este o biblioteca Python minimalista pentru deep learning, care poate rula pe Theano sau TensorFlow. A fost dezvoltat pentru a face dezvoltarea modelelor de deep learning cat mai rapida si usoara posibil pentru cercetare si dezvoltare. Acesta ruleaza pe Python 2.7 sau 3.6 si se poate executa fara probleme pe GPU-uri si procesoare, avand in vedere cadrele de baza. Este eliberat sub licenta permisiva MIT.

Keras a fost dezvoltat si intretinut de François Chollet, un inginer Google, folosind patru principii directoare:

- Modularitate: Un model poate fi inteles doar ca o secventa sau ca un grafic. Toate preocuparile unui model de invatare profunda sunt componente discrete care pot fi combinate in moduri arbitrare.
- Minimalism: Biblioteca ofera suficient pentru a obtine un rezultat, maximizand lizibilitatea.
- Extensibilitate: Componentele noi sunt usor de adaugat si utilizate in mod intentionat in interiorul cadrului, destinate dezvoltatorilor pentru a incerca sa exploreze idei noi.
- Python: Nu exista fisiere model separate cu formate de fisiere personalizate. Totul este Python nativ.

Figura 3.2: Keras

[3]

3.2.3 Numpy

NumPy este biblioteca principala de programare a matricei pentru limbajul Python. Are un rol esential in conductele de analiza a cercetarii in domenii atat de diverse precum fizica, chimia, astronomia, geostiinta, biologia, psihologia, știinta materialelor, inginerie, finante si economie.

Matricea NumPy este o structura de date care stocheaza si acceseaza eficient matrice multidimensionale (cunoscute si sub numele de tensori) si permite o mare varietate de calcule stiintifice. Consta dintr-un pointer catre memorie, impreuna cu metadate folosite pentru a interpreta datele stocate acolo, in special „tipul de date”, „forma” și „pasi”.[4]

Figura 3.3: Numpy

3.2.4 PIL

Python Imaging Library (PIL) este una dintre bibliotecile populare folosite pentru prelucrarea imaginilor. PIL poate fi folosit pentru a afisa imaginea, pentru a crea miniaturi, redimensionare, rotatie, conversie intre formate de fisiere, contrast imbunatatirea, filtrarea si aplicarea altor procesari digitale de imagini tehnici etc. PIL accepta formate de imagine precum PNG, JPEG, GIF, TIFF, BMP etc. Poseda, de asemenea, procesare puternica a imaginii si capabilitati grafice. [7]

Figura 3.4: Python Imaging Library (PIL)

3.2.5 Matplotlib

Matplotlib este un pachet portabil de plotare si imagini 2D in primul rand la vizualizarea datelor stiintifice, de inginerie si financiare. Matplotlib poate fi folosit interactiv din shell-ul Python, numit din scripturi python sau incorporat intr-o aplicatie GUI (GTK, Wx, Tk, Windows). Caracteristicile includ crearea de mai multe axe si cifre pe pagina, navigare interactiva, multe stiluri de linii si simboluri predefinite, imagini, antialiasing, amestecare alfa, date si diagrame financiare, gestionarea fonturilor conform W3C si suport FreeType2, legende si tabele, diagrame pseudocolor, text matematic, samd. Functioneaza atat cu Numarray, cat si cu Numeric. [2]

Figura 3.5: Matplotlib

3.2.6 OpenCV

In acest moment, OpenCV accepta o multime de algoritmi legate de Computer Vision si Machine Learning si se extinde pe zi ce trece. In prezent, OpenCV accepta o mare varietate de limbaje de programare precum C++, Python, Java etc. si este disponibil pe diferite platforme, inclusiv Windows, Linux, OS X, Android, iOS etc. De asemenea, interfete bazate pe CUDA si OpenCL sunt, de asemenea, in curs de dezvoltare activa pentru operatiunile GPU de mare viteza. OpenCV-Python este API-ul Python al OpenCV. Combina cele mai bune calitati ale OpenCV C++ API si limbajul Python . [5]

Figura 3.6: OpenCV

Capitolul 4

Algoritmi

4.1 Convolutional Neural Network (CNN)

Algoritmul utilizat pentru acest proiect este reprezentat de CNN (Convolutional Neural Network). Retelele neuronale au capacitati mai mari de recunoastere a modelelor de imagine si sunt utilizate pe scara larga in algoritmii de computer Vision, iar CNN este o clasa de retele neuronale care se aplica cel mai frecvent la analiza imaginilor vizuale. Clasificarea si detectarea semnelor de circulatie reprezinta sarcini majore in conducerea autonoma, deoarece ofera infomatii despre semnele intalnite si contribuie in luarea deciziilor.

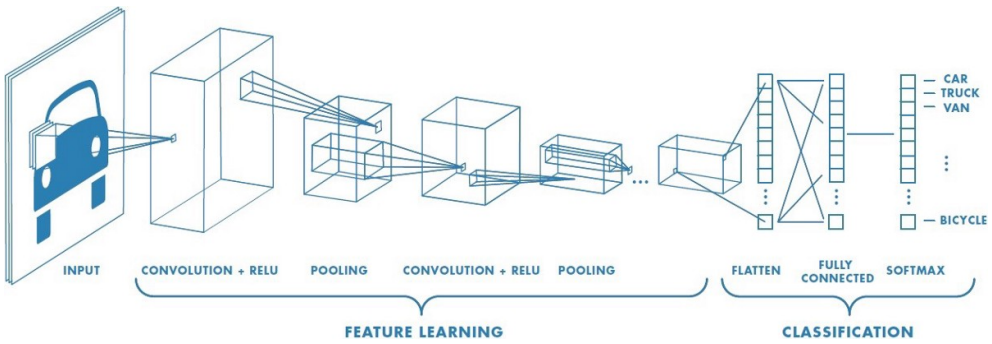

Figura 4.1: Convolutional Neural Network

Imaginea de mai sus arata cum functioneaza un algoritm CNN cu anumite date. Mai intai extrage caracteristicile necesare de date si apoi le clasifica pe baza acelor informatii.

Intrarea unui strat convolutional este o imagine de dimensiunea m $\times$ m $\times$ r, unde m este inaltimea si latimea imaginii, iar r este numarul de canale (ex. imaginile RGB au 3 canale: rosu, verde si albastru).[6]

Asadar, acest domeniu permite masinilor sa vada lumea asa cum o fac oamenii, sa o perceapa intr-un mod similar si chiar sa foloseasca cunostintele pentru o multitudine de sarcini, cum ar fi recunoasterea imaginilor, analiza si clasificarea imaginilor, recrearea media, sistemele de recomandare, procesarea limbajului natural etc. Progresele in viziunea Computer Vision cu invatare profunda au fost construite si perfectionate cu timpul, in primul rand pe un anumit algoritm - o retea neuronala convolutionala .[1]

Capitolul 5

Testare

5.1 Setul de date

Setul de date pentru acest proiect este descarcat de pe site-ul Kaggle. Acesta contine aproximativ 40000 de imagini cu diferite semne de circulatie si este clasificat in 43 de clase diferite. Setul de date este destul de variat, unele dintre clase au mai multe imagini, in timp ce unele au mai putine imagini.

Figura de mai jos cuprinde cele 43 de clase sau imaginile care sunt prezente in setul nostru de date.

Figura 5.1: Imagini din setul de date

5.2 Pregatirea datelor

Pentru a face ca modelul sa fie usor de inteles si sa prezica un rezultat precis, mai intai am efectuat cateva operatii asupra setului nostru de date .

Astfel, am efectuat urmatorii pasi:

- Am importat setul de date impreuna cu bibliotecile necesare, iar apoi am creat un dictionar unde am salvat denumirile tuturor semnelor de circulatie aflate in setul nostru de date (figura 5.2). In continuare, am extras numarul de imagini pentru fiecare clasa in parte pentru a sti care este volumul de date cu care lucram pentru fiecare categorie (figura 5.3).

  Dupa colectarea datelor, am creat un array cu toate imaginile pe care le-am redimensionat(30x30, 'rgb') si i-am atribuit fiecareia eticheta corespunzatoare (Denumirea semnului de circulatie ilustrat de aceasta) (vezi Figura 5.4);

```
semne_circulatie = { 0:'Limita de viteza (20km/h)',
            1:'Limita de viteza (30km/h)',
            2:'Limita de viteza (50km/h)',
            3:'Limita de viteza (60km/h)',
            4:'Limita de viteza (70km/h)',
            5:'Limita de viteza (80km/h)',
            6:'Sfaritul zonei cu limita 80km/h',
            7:'Limita de viteza (100km/h)',
            8:'Limita de viteza (120km/h)',
            9:'Trecerea interzisa',
            10:' Accesul interzis vehiculelor cu peste 3.5 tone',
            11:'Prioritate',
            12:'Drum cu prioritate',
            13:'Cedeaza trecerea',
            14:'Stop',
            15:'Accesul interzis vehiculelor',
            16:'Depasirea interzisa a autovehiculelor cu peste 3.5 tone',
            17:'Intrarea interzisa',
            18:'Atentie sporita',
            19:'Curba periculoasa la stanga',
            20:'Curba periculoasa la dreapta',
            21:'Curba dubla',
            22:'Drum cu denivelari',
            23:'Drum alunecos',
            24:'Drumul se ingusteaza pe dreapta',
            25:'Drum in lucru',
            26:'Semafor',
            27:'Pietoni',
            28:'Copii care traversează',
            29:'Traversarea bicicletelor',
            30:'Atenție la gheață/zăpadă',
            31:'Atentie animale',
            32:'Sfarsit restrictii',
            33:'Virare la dreapta',
            34:'Virare la stanga',
            35:'Doar înainte',
            36:'In fata sau la dreapa',
            37:'In fata sau la stanga',
            38:'La dreapta',
            39:'La stanga',
            40:'Sens giratoriu',
            41:'Sfarsitul zonei unde depasirea e interzisa',
            42:'Sfarsitul interzicerii trecerii vehiculelor de peste 3.5 tone' }
```

Figura 5.2: Dictionarul cu semnele de circulatie

```
folders = os.listdir(train_path)

numar_imagini_antrenare = []
nume_semne_circulatie = []

for folder in folders:
    fisiere_antrenare = os.listdir(train_path + '/' + folder)
    numar_imagini_antrenare.append(len(fisiere_antrenare))
    nume_semne_circulatie.append(semne_circulatie[int(folder)])

# Sortarea setului de date pe baza numărului de imagini din fiecare clasă
zipped_lists = zip(numar_imagini_antrenare, nume_semne_circulatie)
perechi_sortate = sorted(zipped_lists)
tuples = zip(*perechi_sortate)
numar_imagini_antrenare, nume_semne_circulatie = [list(tuple) for tuple in tuples]

# Trasarea numărului de imagini din fiecare clasă
plt.figure(figsize=(21, 10))
plt.bar(nume_semne_circulatie, numar_imagini_antrenare)
plt.xticks(nume_semne_circulatie, rotation='vertical')
plt.show()
```

Figura 5.3: Extragerea numarului de imagini

```
img_data = []
img_etichete = []

for i in range(NUMAR_SEMNE):
    path = data_dir + '\\Train\\' + str(i)
    imagini = os.listdir(path)

    for img in imagini:
        try:
            image = cv2.imread(path + '\\' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((inaltime_IMG, latime_IMG))
            img_data.append(np.array(resize_image))
            img_etichete.append(i)
        except:
            print("Error in " + img)

# Schimbarea listei în matricea numpy
img_data = np.array(img_data)
img_etichete = np.array(img_etichete)

print(img_data.shape, img_etichete.shape)
```

Figura 5.4: Pregatirea datelor

- am amestecat datele care urmeaza sa fie antrenate si le-am impartit in 2 categorii: de antrenare si de validare (vezi Figura 5.7);

```
shuffle_indexes = np.arange(img_data.shape[0])
np.random.shuffle(shuffle_indexes)
img_data = img_data[shuffle_indexes]
img_etichete = img_etichete[shuffle_indexes]
```

```
X_train, X_val, y_train, y_val = train_test_split(img_data, img_etichete, test_size=0.3, random_state=42, shuffle=True)

X_train = X_train/255 # normalizarea datelor
X_val = X_val/255

print("X_train.shape", X_train.shape)
print("X_valid.shape", X_val.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_val.shape)
```

```
X_train.shape (27446, 30, 30, 3)
X_valid.shape (11763, 30, 30, 3)
y_train.shape (27446,)
y_valid.shape (11763,)
```

Figura 5.5: Amestecarea datelor si sortarea acestora

- am creat un model utilizat pentru antrenarea algoritmului de machine learning CNN, astfel incat sa obtinem un rezultat cat mai precis (o acuratete de 98 la suta) (vezi Figura 5.7).

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(inaltime_IMG,latime_IMG,canale_IMG)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense(43, activation='softmax')
])
```

Figura 5.6: Structurarea modelului

```
aug = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.15,
    horizontal_flip=False,
    vertical_flip=False,
    fill_mode="nearest")

history = model.fit(aug.flow(X_train, y_train, batch_size=32), epochs=epochs, validation_data=(X_val, y_val))
```

Figura 5.7: Antrenarea setului de date

Ulterior, am folosit modelul respectiv pentru realizarea predictiilor (figura 5.8).

```
plt.figure(figsize = (25, 25))

start_index = 0
for i in range(20):
    plt.subplot(10, 2, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = pred[start_index + i]
    actual = labels[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(semne_circulatie[actual], semne_circulatie[prediction]), color = col)
    plt.imshow(X_test[start_index + i])
plt.show()
```

Figura 5.8: Predictii

Capitolul 6

Rezultate

```
Epoch 1/10
858/858 [==============================] - 58s 60ms/step - loss: 1.0678 - accuracy: 0.7157 - val_loss: 0.0862 - val_accuracy: 0.9716
Epoch 2/10
858/858 [==============================] - 48s 56ms/step - loss: 0.1737 - accuracy: 0.9465 - val_loss: 0.0290 - val_accuracy: 0.9918
Epoch 3/10
858/858 [==============================] - 47s 55ms/step - loss: 0.1019 - accuracy: 0.9690 - val_loss: 0.0228 - val_accuracy: 0.9926
Epoch 4/10
858/858 [==============================] - 49s 57ms/step - loss: 0.0693 - accuracy: 0.9791 - val_loss: 0.0154 - val_accuracy: 0.9954
Epoch 5/10
858/858 [==============================] - 49s 57ms/step - loss: 0.0597 - accuracy: 0.9817 - val_loss: 0.0131 - val_accuracy: 0.9969
Epoch 6/10
858/858 [==============================] - 48s 56ms/step - loss: 0.0450 - accuracy: 0.9858 - val_loss: 0.0114 - val_accuracy: 0.9964
Epoch 7/10
858/858 [==============================] - 47s 55ms/step - loss: 0.0383 - accuracy: 0.9878 - val_loss: 0.0187 - val_accuracy: 0.9940
Epoch 8/10
858/858 [==============================] - 47s 55ms/step - loss: 0.0327 - accuracy: 0.9898 - val_loss: 0.0056 - val_accuracy: 0.9985
Epoch 9/10
858/858 [==============================] - 48s 56ms/step - loss: 0.0227 - accuracy: 0.9923 - val_loss: 0.0066 - val_accuracy: 0.9984
Epoch 10/10
858/858 [==============================] - 49s 58ms/step - loss: 0.0269 - accuracy: 0.9915 - val_loss: 0.0049 - val_accuracy: 0.9991
```

Figura 6.1: Rezultatele din urma antrenarii modelului

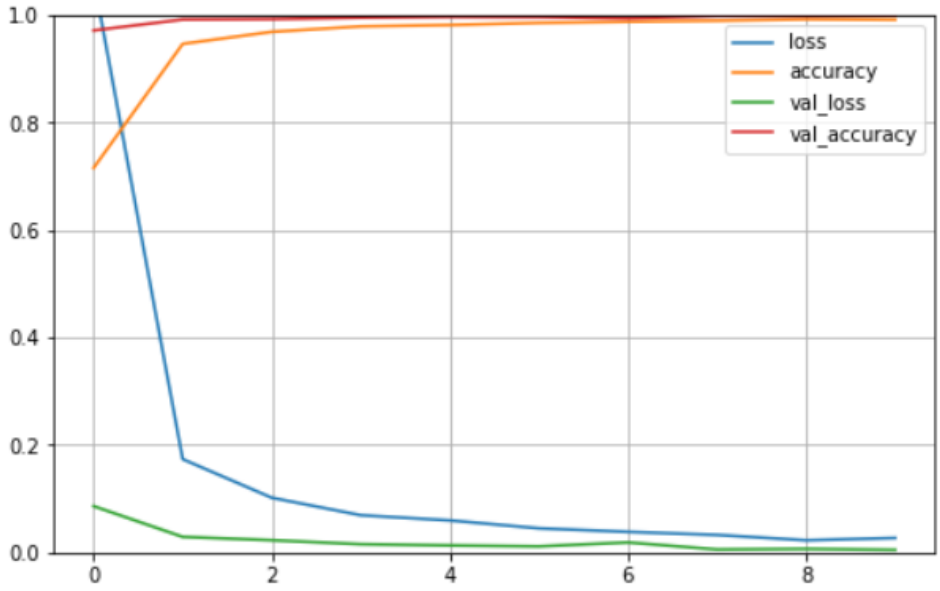

Figura 6.2: Graficul de precizie al modelului antrenat in raport cu nr. de epoci dat (10)

Pentru a verifica eficienta programului creat, am facut o serie de teste folosindu-ne de setul de date utilizat.

Mai intai am testat pe un volum mai mic de imagini, mai exact 10, iar rezultatele sunt prezentate in figura de mai jos.

Figura 6.3: Rezultate pentru un set de 10 imagini

Figura 6.4: Rezultate pentru un set de 25 imagini

Figura 6.5: Rezultate pentru un set de 50 imagini

Figura 6.6: Rezultate pentru un set de 100 imagini

Figura 6.7: Rezultate pentru un set de 100 imagini

Putem observa cu usurinta privind figurile, ca exista o precizie aproape perfecta a algoritmului de detectare a semnelor de circulatie, existand o singura predictie gresita aparuta dupa multe teste cu rezultate impecabile.

De mentionat este si faptul ca algoritmul nu recunoaste decat semne de circulatie din setul nostru de date.

Bibliografie

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[2] Paul Barrett, John Hunter, J Todd Miller, J-C Hsu, and Perry Greenfield. matplotlib–a portable python plotting package. In *Astronomical data analysis software and systems XIV*, volume 347, page 91, 2005.

[3] Jason Brownlee. *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.

[4] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[5] Alexander Mordvintsev and K Abid. Opencv-python tutorials documentation. *Obtenido de https://media. readthedocs. org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals. pdf*, 2014.

[6] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[7] P Umesh. Image processing in python. *CSI Communications*, 23(2), 2012.