PROCESIRANJE X-ZRAKA, PRIMJENA U DIJAGNOSTICIRANJU VIRUSA COVID-19

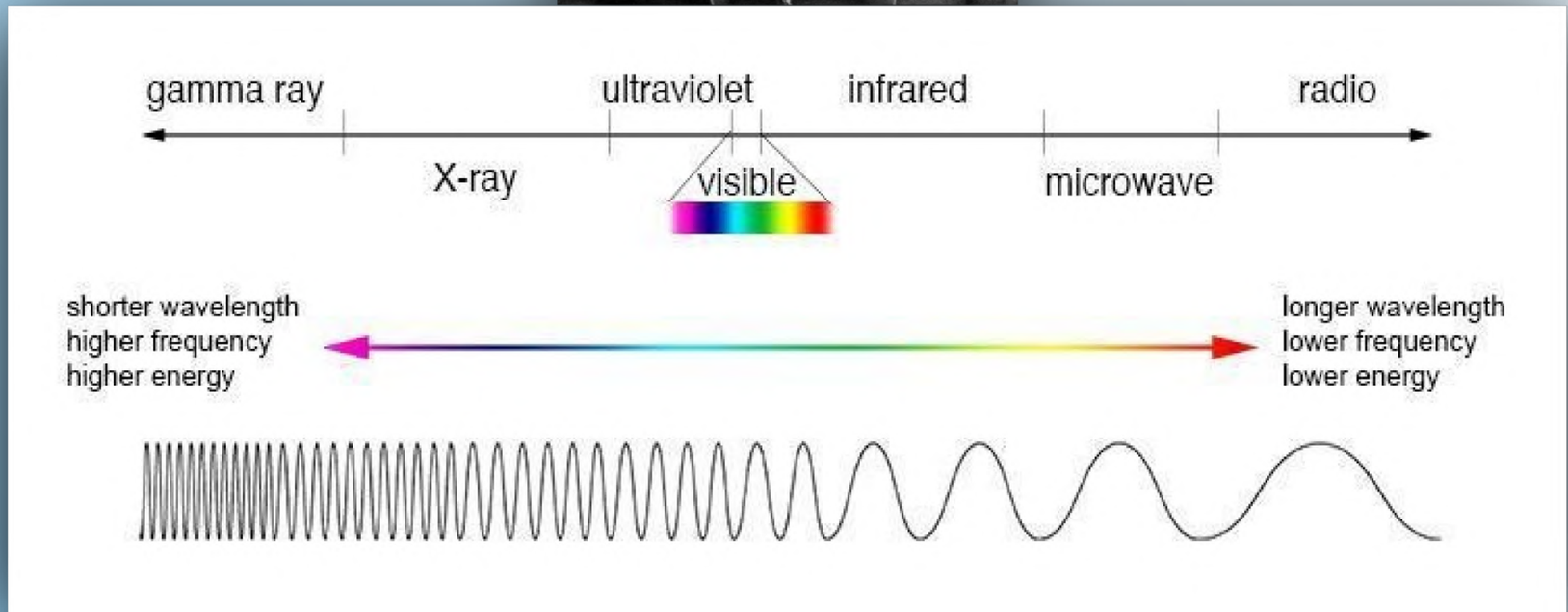
MASNOPITA MUHAMED I
MATORUGA FARIS

UVOD

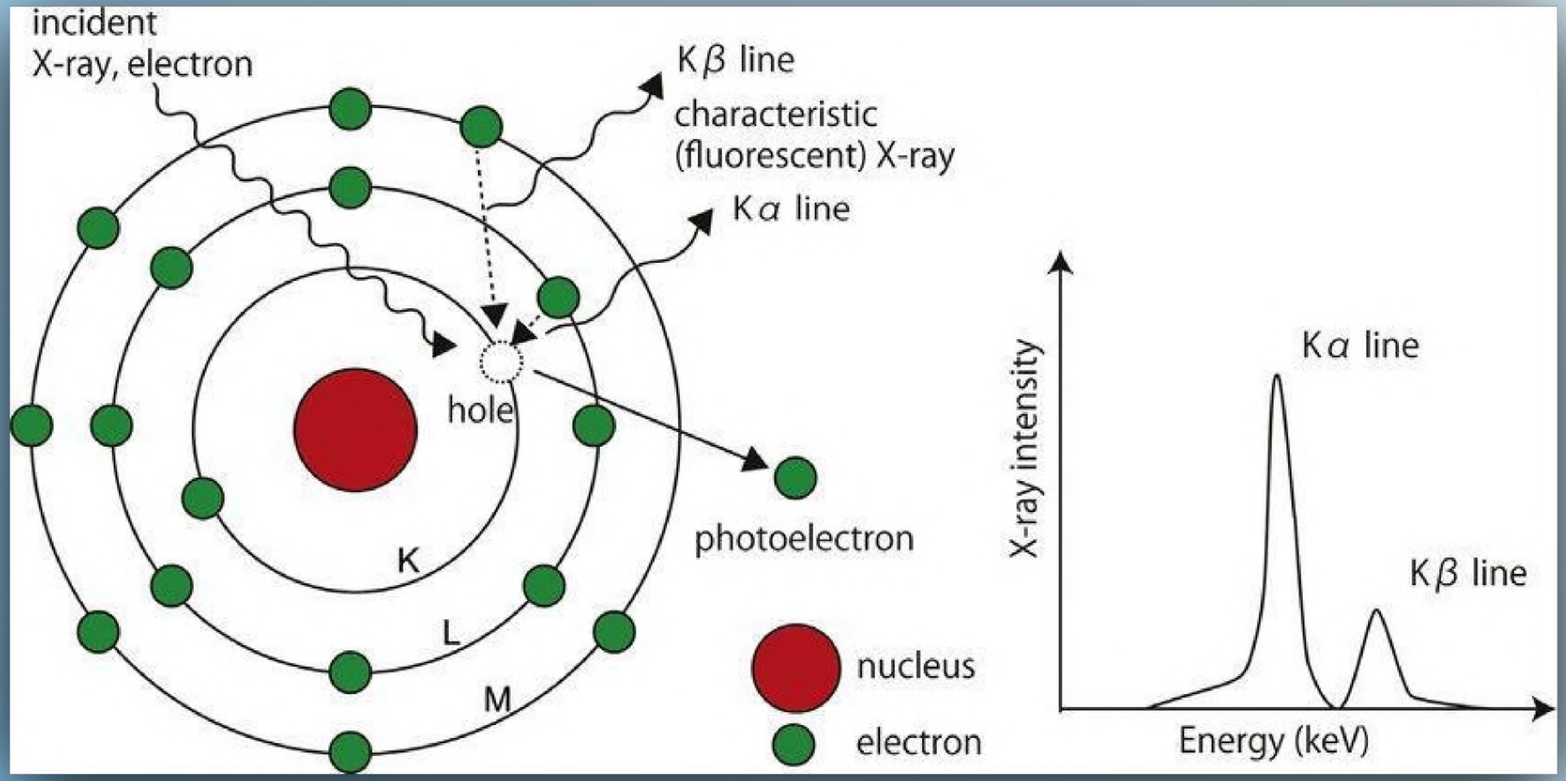
Tehnologija X-zraka je veoma važno otkriće
Nevidljivo pretvara u vidljivo
CT

X-ZRAKE OSNOVE

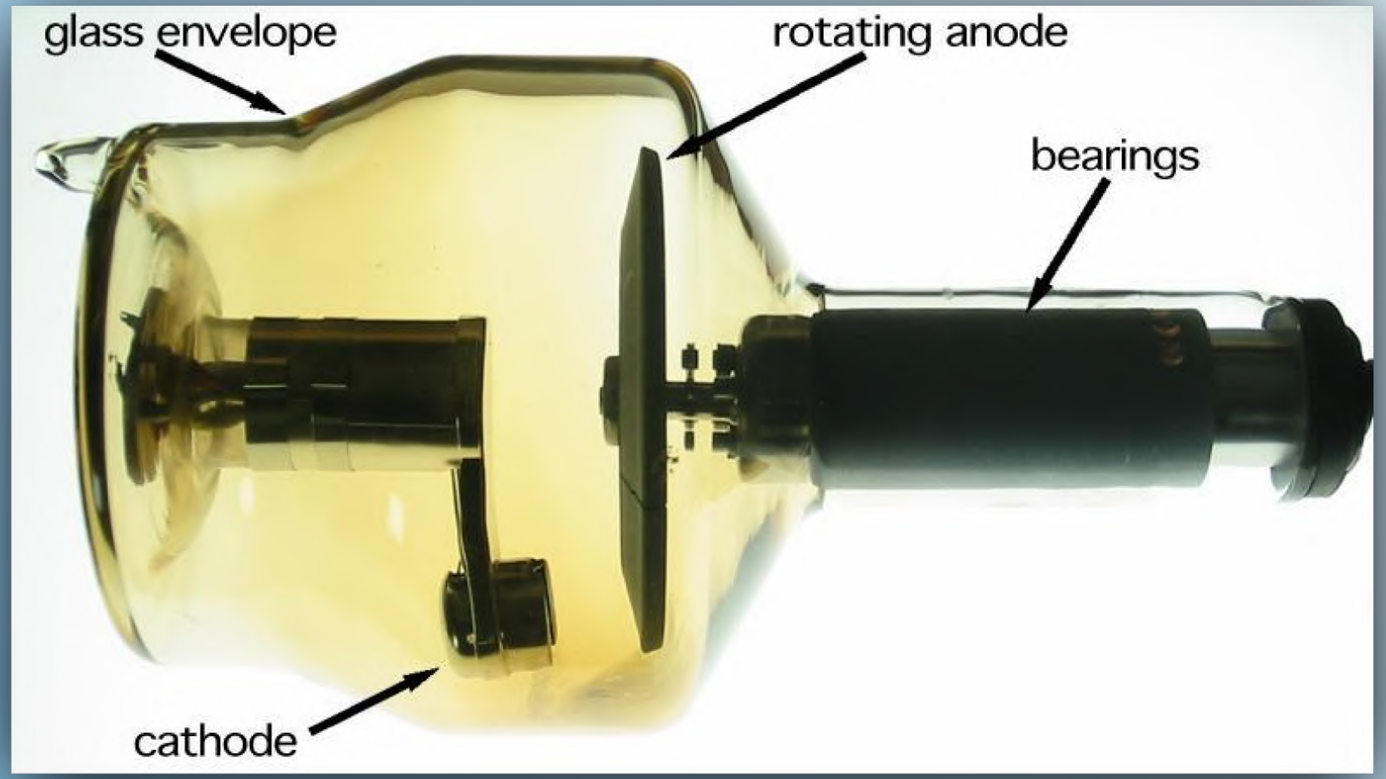
1895. otkrio Wilhelm Conrad Röntgen
1901. Nobelova nagrada
Radiografija
Spektar zraka
Rendgenski radiograf
Algoritam za rekonstrukciju slike



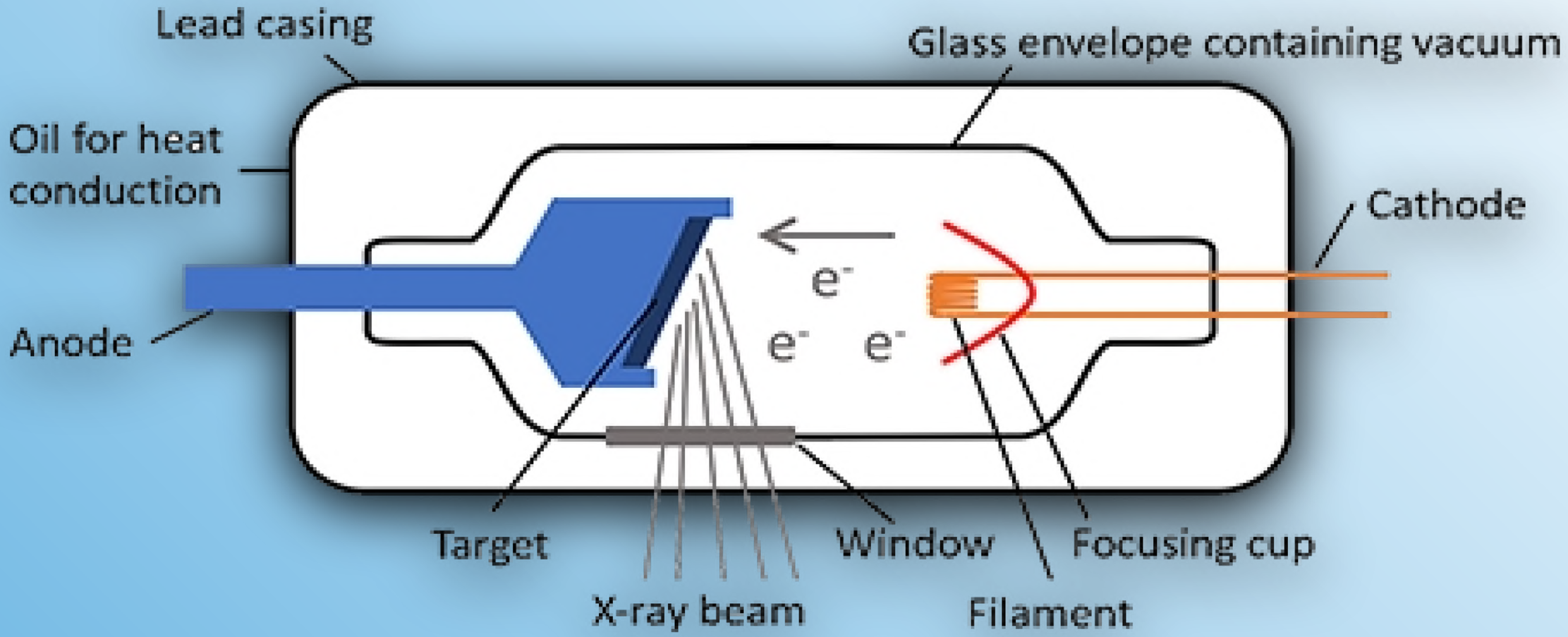
GENERACIJA X-ZRAKA



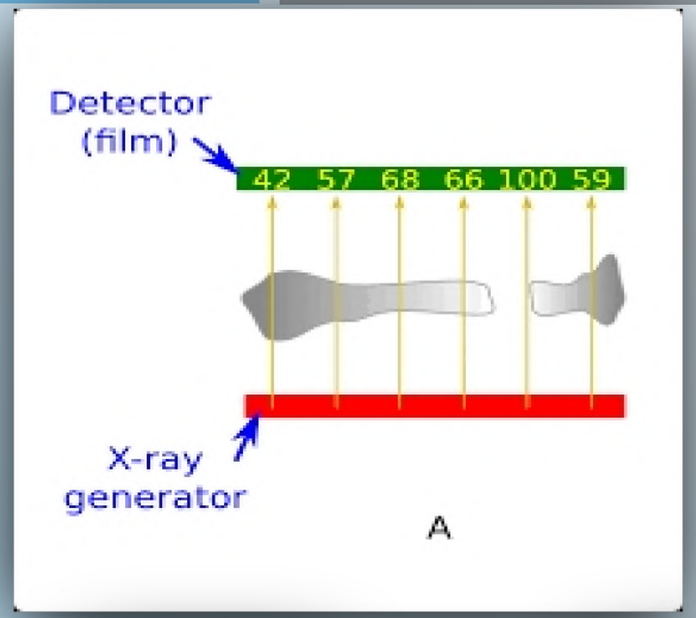
ROTACIONA ANODA

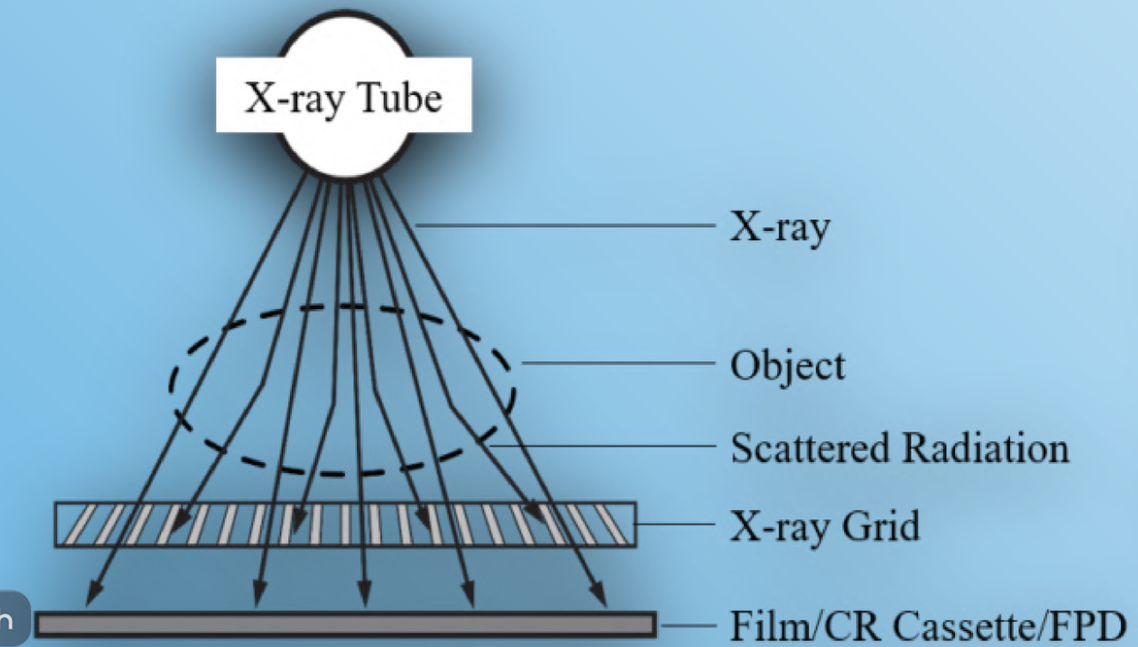


RENDGENSKA CIJEV

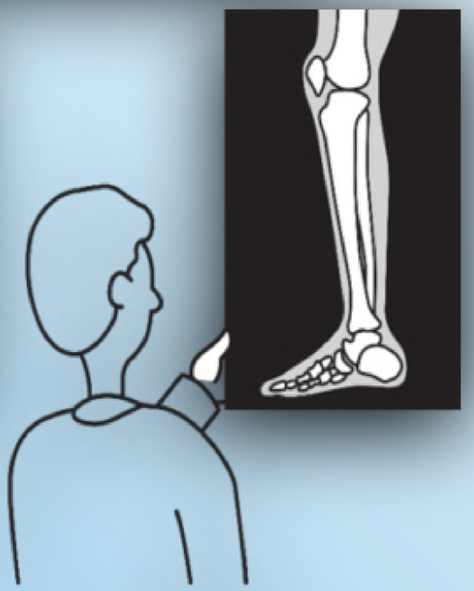


2D PROJEKCIJA X-ZRAKA





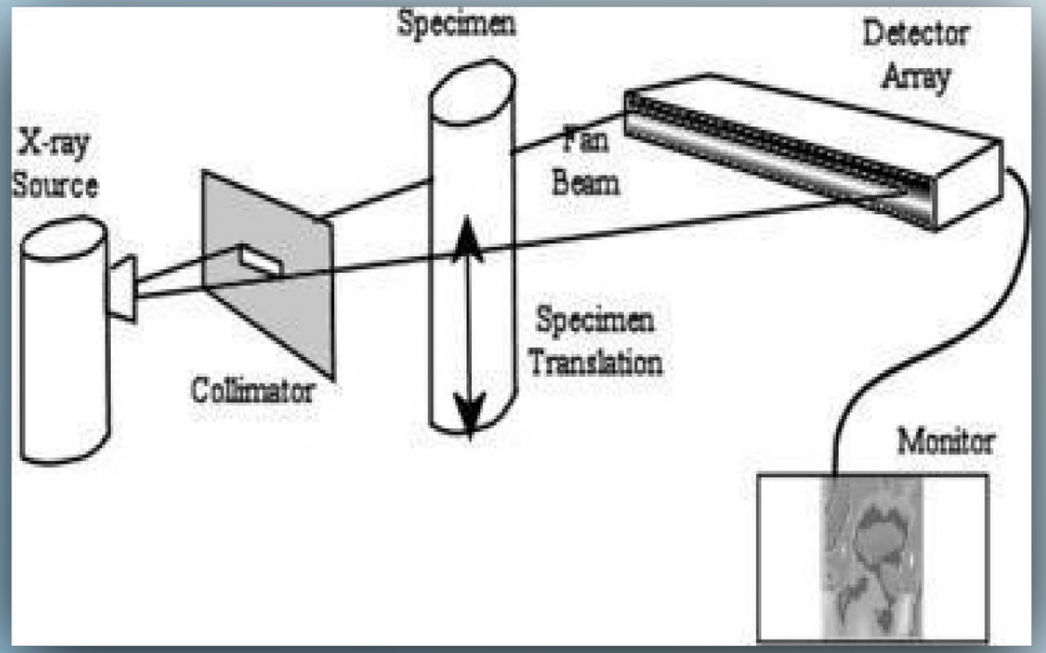
Without Grid



With Grid

RADIOGRAFIJA U REALNOM VREMENU

brzina pregleda usavršena, kvalitet slika veći, oprema koja se koristi, metode za analizu i skladištenje relativno više koštaju

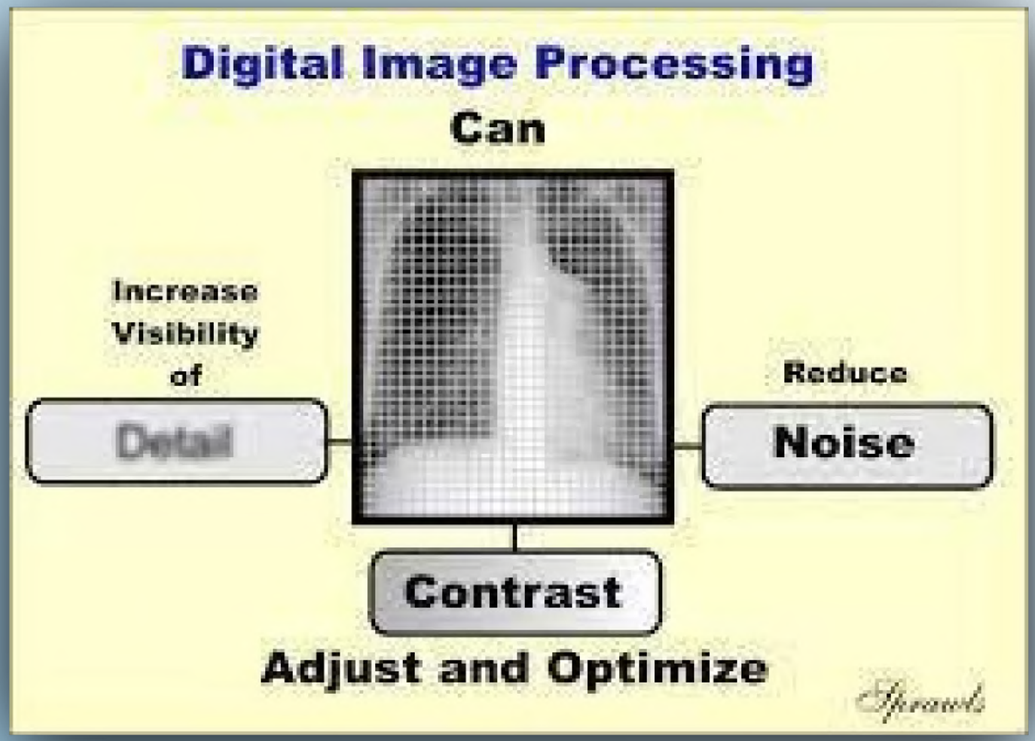


PROCESIRANJE RENDGENSKI DIGITALNIH SLIKA

smanjenje šuma na slici.

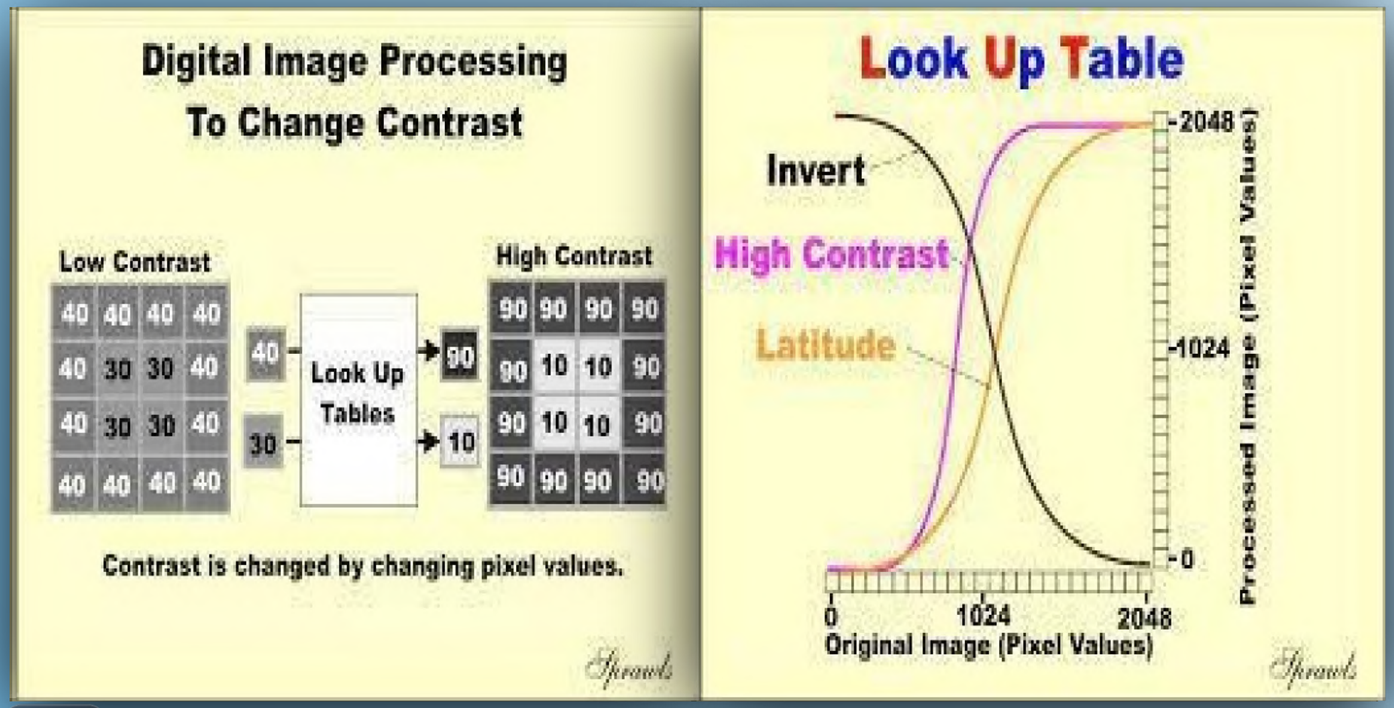
povećanje vidljivosti slike na detalje,

regulisanje i optimizacija kontrasta slike.

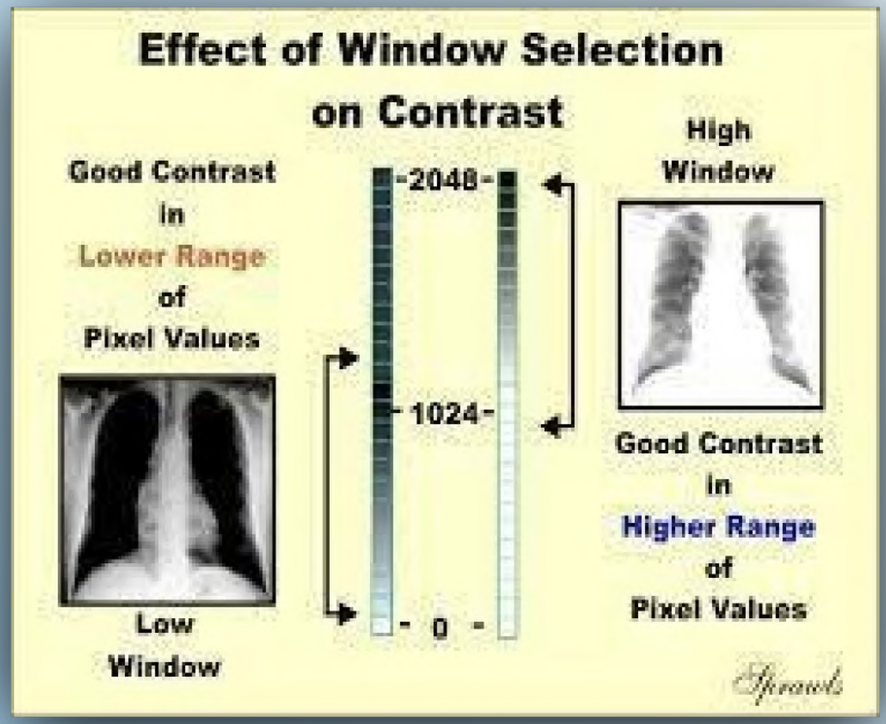


REGULISANJE KONTRASTA RENDGENSKIH SLIKA

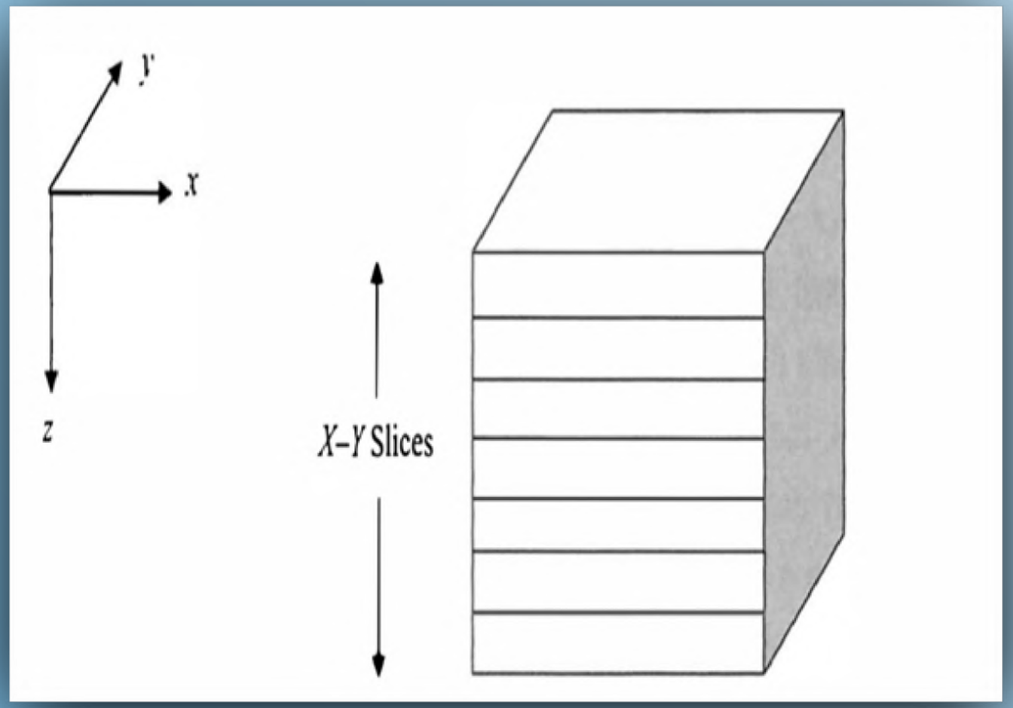
Look Up Table (LUT) procesiranje

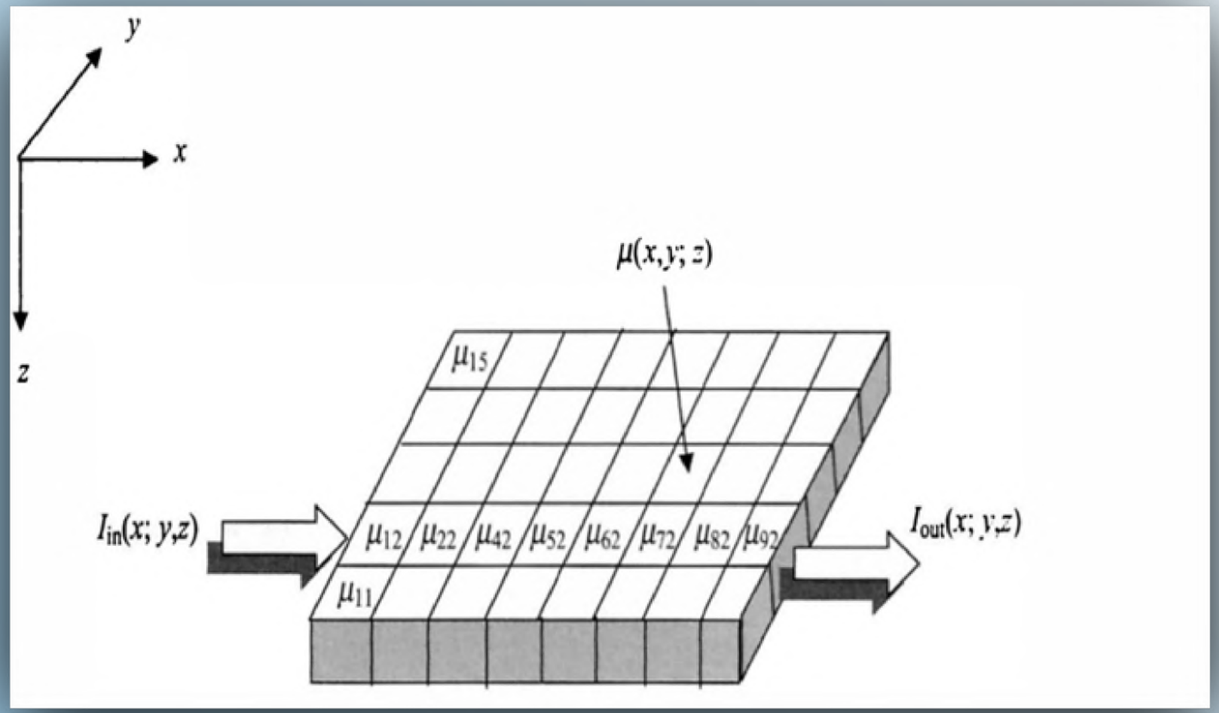


Windowing procesiranje

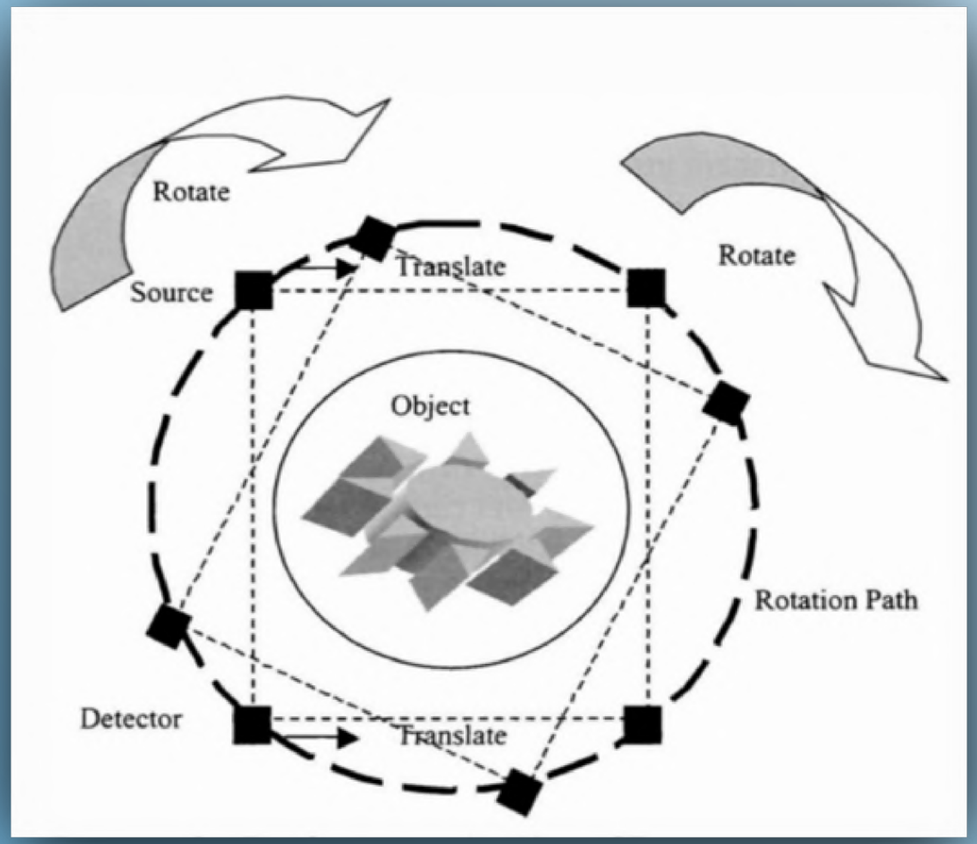


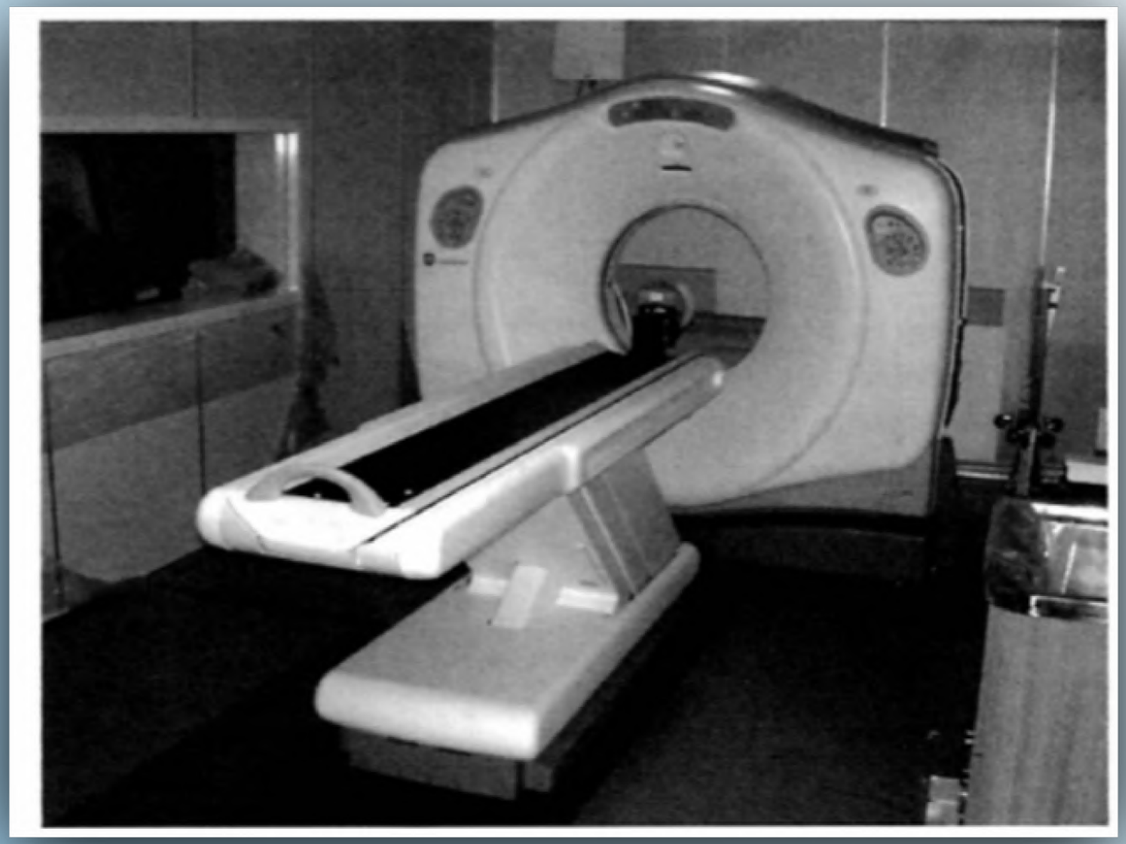
CT X-ZRAKA (1)



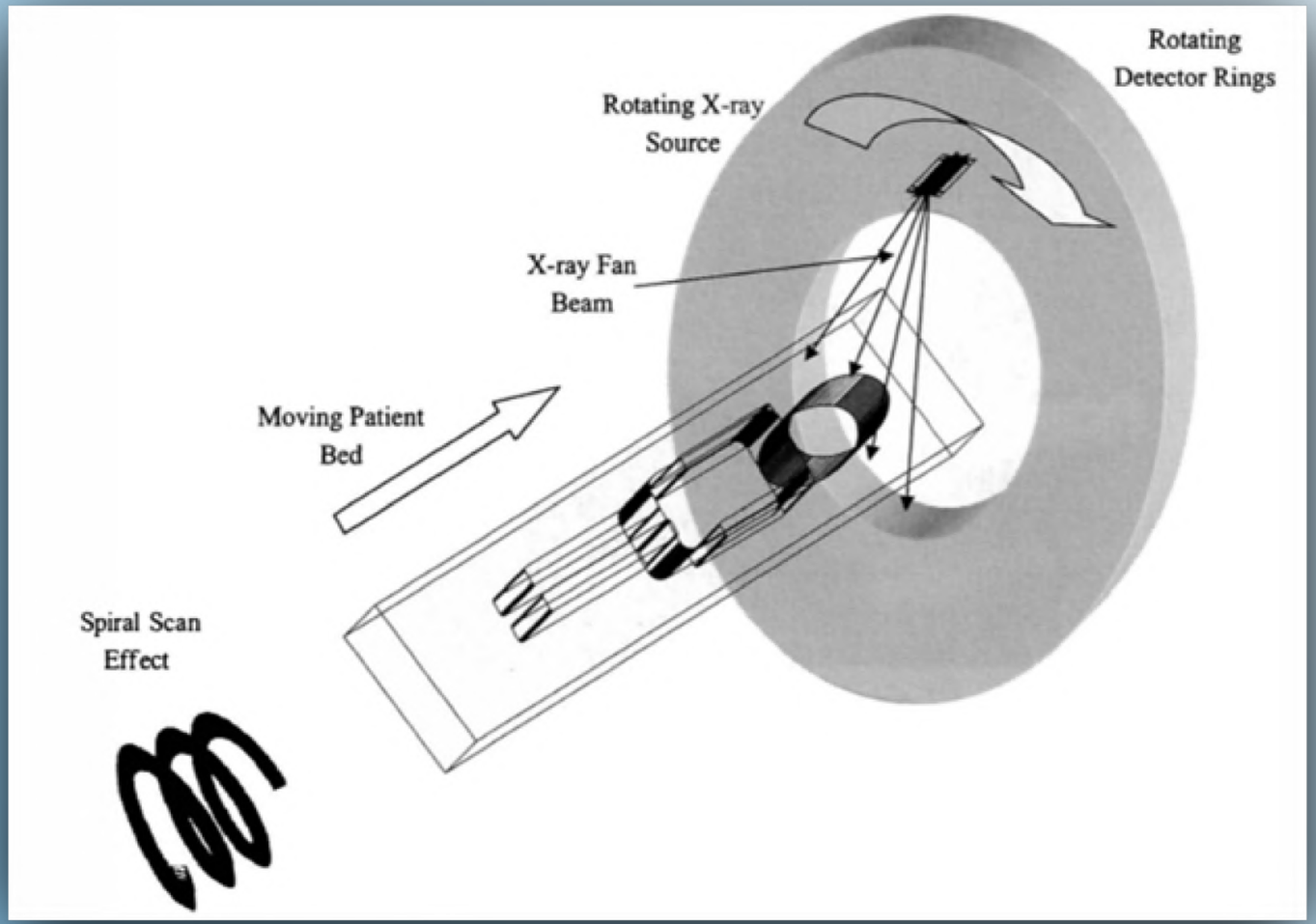


CT X-ZRAKA (2)

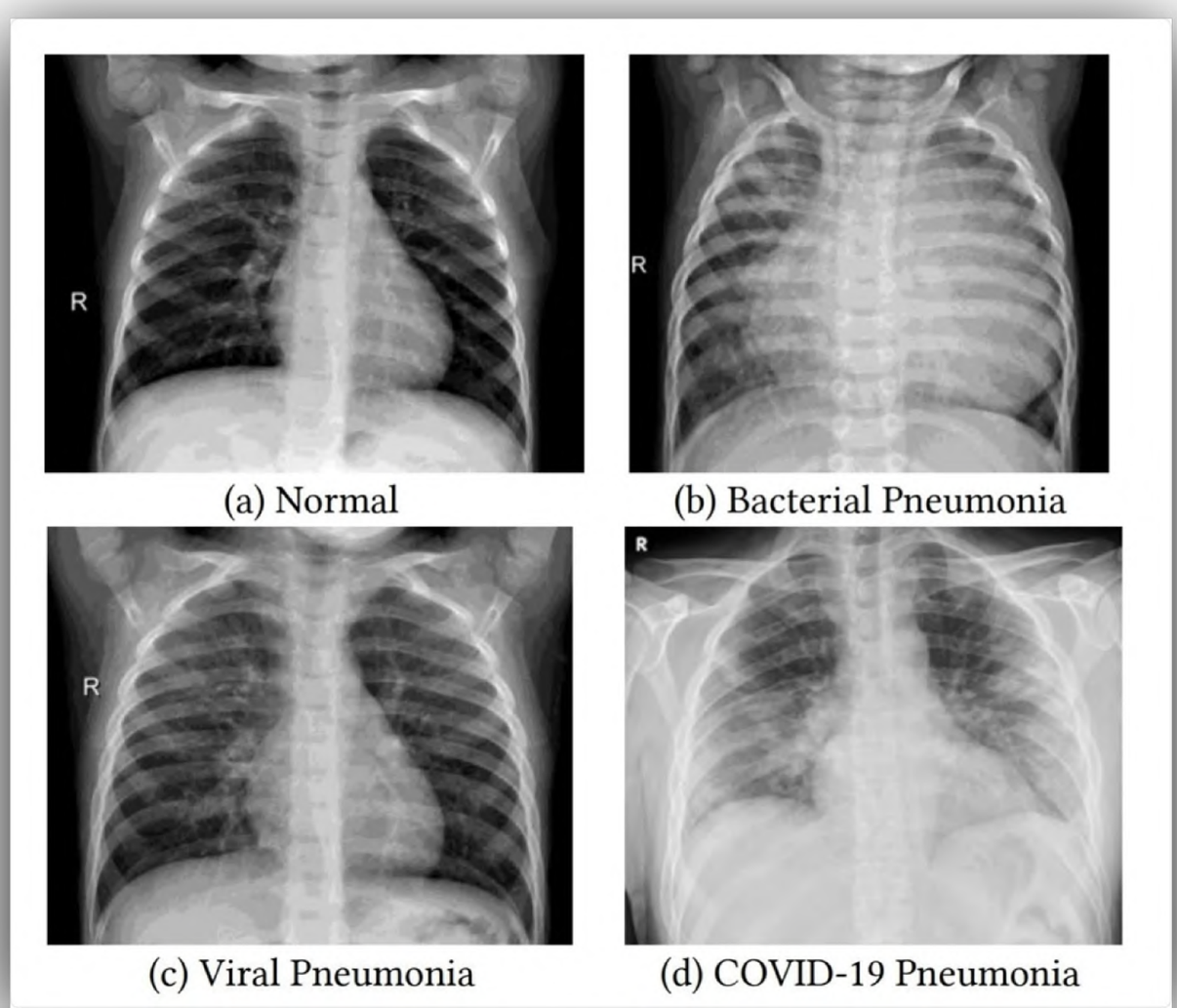

Rotate
Translate
Rotate
Source
Object
Rotation Path
Detector
Translate



SPIRALNI CT


Rotating
Detector Rings
Rotating X-ray
Source
X-ray Fan
Beam
Moving Patient
Bed
Spiral Scan
Effect

PRIMJENA PROCESIRANJA X-ZRAKA U DIJAGNOSTICIRANJU VIRUSA COVID-19



AUTOMATIZACIJA PREPOZNAVANJA BOLESTI KORISTEĆI PHYTON

- PyCharm
- phyton
- Dvije klase:
- train_covid19.py – klasa u kojoj testiramo sve(nešto kao main),
- build_covid_dataset.py – konekcija s bazom podataka.

TRAIN_COVID19.PY (1)

- From tenserflow.keras.preprocessing.image import ImageDataGenerator (za generisanje serije podataka tenzorske slike uz povećanje podataka u realnom vremenu),
- From tenserflow.keras.applications import VGG16 (za instantaciju modela VGG16[1]),
- From tenserflow.keras.layers import AveragePooling2D (za operacije udruživanja za prostorne podatke),
- From tenserflow.keras.layers import Dropout (primjenjuje Dropout/napuštanje na ulaz - Sloj Dropout nasumično postavlja ulazne jedinice na 0 s frekvencijom stope u svakom koraku tokom vremena treninga, što pomaže u sprječavanju prekomjernog prilagođavanja),

TRAIN_COVID19.PY (2)

- From tenserflow.keras.layers import Flatten (Izjednačava unos. Ne utiče na veličinu serije),
- From tenserflow.keras.layers import Dense (uobičajeni gusto povezani NN sloj),
- From tenserflow.keras.layers import Input (za instanciranje Keras tenzora),
- From tenserflow.keras.models import Model (grupira slojeve u objekt sa karakteristikama obuke i zaključivanja),
- From tenserflow.keras.optimizers import Adam (optimizator koji implementira Adam algoritam[2]),
- From tenserflow.keras.utils import to_categorical (za pretvorbu cijelih brojeva u binarnu matricu klase),
- From sklearn.preprocessing import LabelBinarizer (LabelBinarizer olakšava proces konverzije višeklasne oznake u binarne oznake pomoću metode transformacije),
- From sklearn.model_selection import train_test_split (za dijeljenje nizova/matrica u nasumične testne podskupove),
- From sklearn.metrics import classification_report (izrađuje tekstualni izvještaj koji prikazuje glavne metrike klasifikacije),
- From sklearn.metrics import confusion_matrix (izračunava matricu konfuzije za projenu tačnosti klasifikacije),
- From imutils import paths (uvodimo zbog putanje slika u memoriji),
- import matplotlib.pyplot import plt (funkcija za rad s figurama),
- import numby as np (za naučno računanje),
- import argparse (olakšava pisanje korisničkih interfejsa),
- Import Cv2 (modul za Phytonov web-paket),
- Import os (modul izvozi skup učinkovitih funkcija).

TRAIN_COVID19.PY (3)

```
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
   help=r"C:\Users\masno\Desktop\MuhamedFaks\IV semestar\Digitalno procesiranje signala\Procesiranje X-ray zraka\dataset\dataset\covid") #path to input
dataset
ap.add_argument("-p", "--plot", type=str, default="plot.png",
   help=r"C:\Users\masno\Desktop\MuhamedFaks\IV semestar\Digitalno procesiranje signala\Procesiranje X-ray zraka\dataset\dataset\plots") #path to output
loss/accuracy plot
ap.add_argument("-m", "--model", type=str, default="covid19.model",
   help=r"C:\Users\masno\Desktop\MuhamedFaks\IV semestar\Digitalno procesiranje signala\Procesiranje X-ray zraka\dataset\dataset\plots") #path to output
loss/accuray plot
args = vars(ap.parse_args())
```

TRAIN_COVID19.PY (4)

```
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []
```

```
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]

    # load the image, swap color channels, and resize it to be a fixed
    # 224x224 pixels while ignoring aspect ratio
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))

    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)
```

TRAIN_COVID19.PY (5)

```
# convert the data and labels to NumPy arrays while scaling the pixel
# intensities to the range [0, 255]
data = np.array(data) / 255.0
labels = np.array(labels)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
```

```
# load the VGG16 network, ensuring the head FC layer sets are left
# off
baseModel = VGG16(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

TRAIN_COVID19.PY (6)

```
# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
```

```
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

```
# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)
```

```
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
	layer.trainable = False
```

TRAIN_COVID19.PY (7)

```
rint("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
  metrics=["accuracy"])
```

```
print("[INFO] training head...")
H = model.fit_generator(
  trainAug.flow(trainX, trainY, batch_size=BS),
  steps_per_epoch=len(trainX) // BS,
  validation_data=(testX, testY),
  validation_steps=len(testX) // BS,
  epochs=EPOCHS)
```

TRAIN_COVID19.PY (8)

```
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
```

```
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
   target_names=lb.classes_))

# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
```

TRAIN_COVID19.PY (9)

```
# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

BUILD_COVID_DATASET.PY (1)

```
# import the necessary packages
import pandas as pd
import argparse
import shutil
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--covid", required=True,
   help="C:\Users\masno\Desktop\MuhamedFaks\IV semestar\Digitalno procesiranje signala\Procesiranje X-ray
zraka\dataset\dataset\covid") #path to base directory for COVID-19 dataset
ap.add_argument("-o", "--output", required=True,
   help="C:\Users\masno\Desktop\MuhamedFaks\IV semestar\Digitalno procesiranje signala\Procesiranje X-ray
zraka\dataset\dataset\normal") #path to directory where 'normal' images will be stored
args = vars(ap.parse_args())

# construct the path to the metadata CSV file and load it
csvPath = os.path.sep.join([args["covid"], "metadata.csv"])
df = pd.read_csv(csvPath)
```

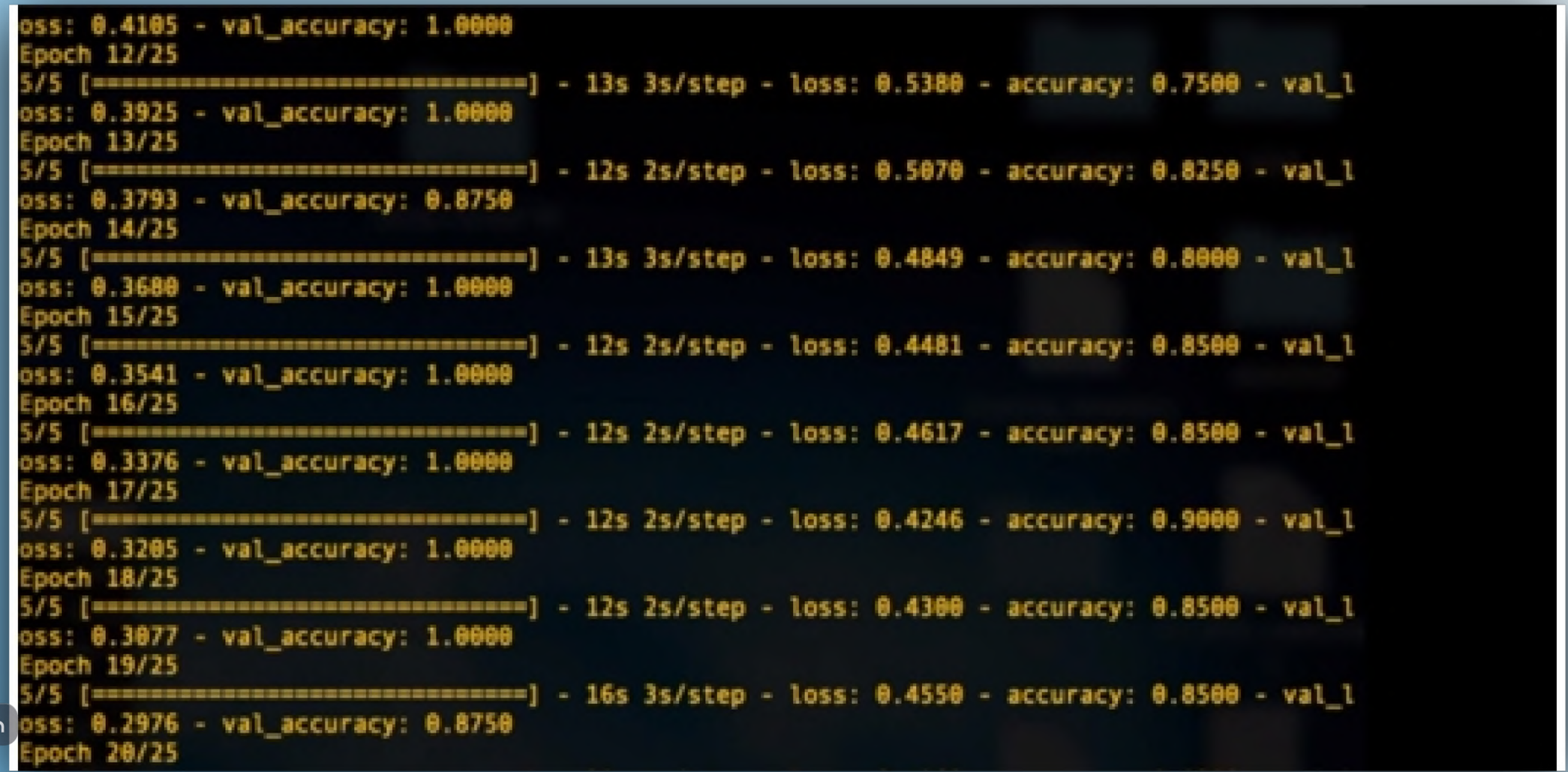
BUILD_COVID_DATASET.PY (2)

```
# if the input image file does not exist (there are some errors in
# the COVID-19 metadeta file), ignore the row
if not os.path.exists(imagePath):
	continue

# extract the filename from the image path and then construct the
# path to the copied image file
filename = row["normal"].split(os.path.sep)[-1]
outputPath = os.path.sep.join([args["plot"], filename])

# copy the image
shutil.copy2(imagePath, outputPath)
```

PRIKAZ REZULTATA (1)

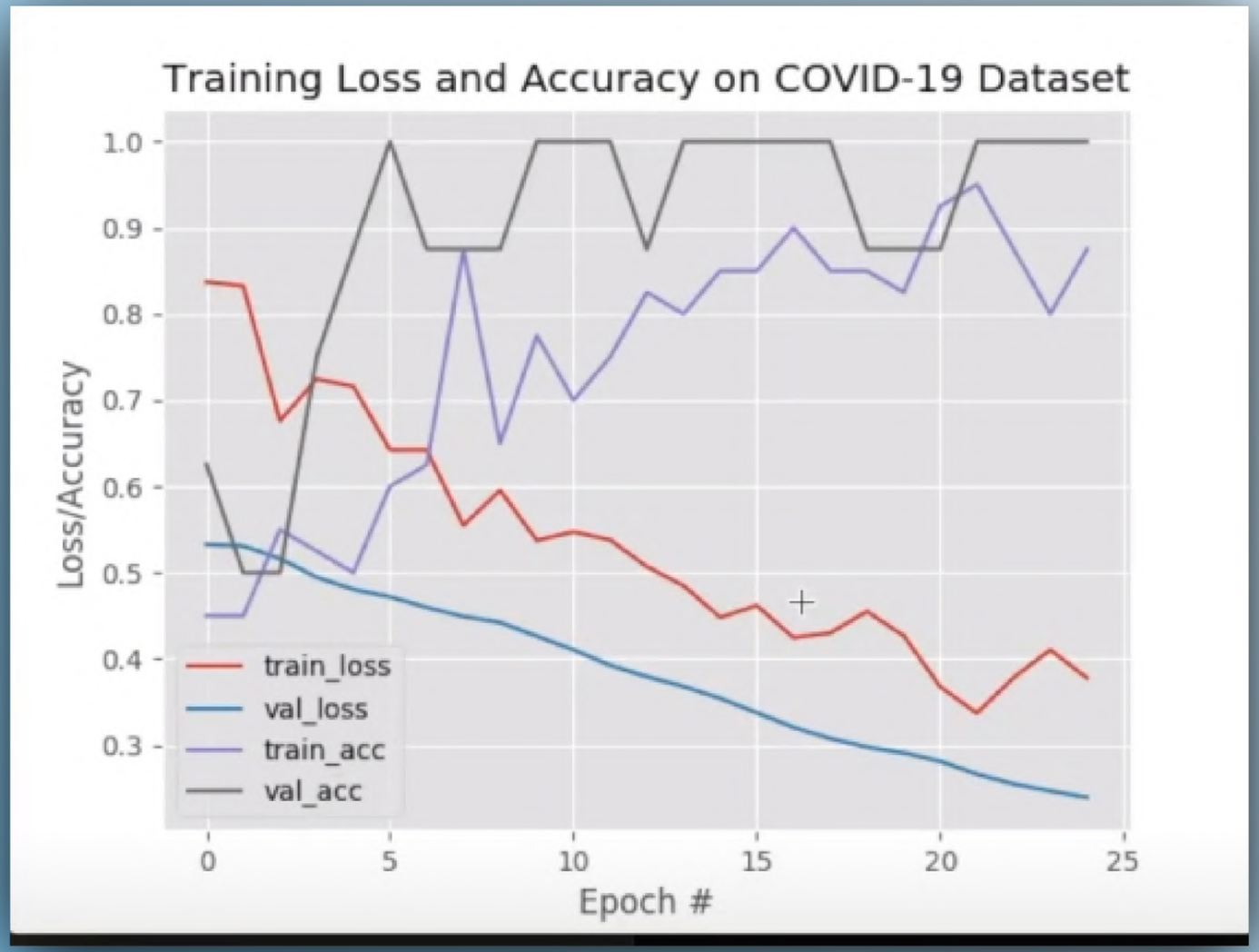


PRIKAZ REZULTATA (2)

```
5/5 [==============================] - 16s 3s/step - loss: 0.4550 - accuracy: 0.8500 - val_l
oss: 0.2976 - val_accuracy: 0.8750
Epoch 20/25
5/5 [==============================] - 20s 4s/step - loss: 0.4269 - accuracy: 0.8250 - val_l
oss: 0.2909 - val_accuracy: 0.8750
Epoch 21/25
5/5 [==============================] - 21s 4s/step - loss: 0.3677 - accuracy: 0.9250 - val_l
oss: 0.2813 - val_accuracy: 0.8750
Epoch 22/25
5/5 [==============================] - 14s 3s/step - loss: 0.3371 - accuracy: 0.9500 - val_l
oss: 0.2663 - val_accuracy: 1.0000
Epoch 23/25
5/5 [==============================] - 15s 3s/step - loss: 0.3783 - accuracy: 0.8750 - val_l
oss: 0.2550 - val_accuracy: 1.0000
Epoch 24/25
5/5 [==============================] - 12s 2s/step - loss: 0.4104 - accuracy: 0.8000 - val_l
oss: 0.2467 - val_accuracy: 1.0000
Epoch 25/25
5/5 [==============================] - 13s 3s/step - loss: 0.3778 - accuracy: 0.8750 - val_l
oss: 0.2393 - val_accuracy: 1.0000
[INFO] evaluating network...
              precision    recall  f1-score   support

       covid       1.00      0.00      0.00         5
      normal       0.83      1.00      0.91         5
```

GRAFIČKI PRIKAZ TAČNOSTI ANALIZE



ZAKLJUČAK

Opisani alat se veoma dobro može koristiti za naprednije primjene.

HVALA NA PAŽNJI