MACHINE LEARNING & SMART SYSTEMS

MLSS A-1

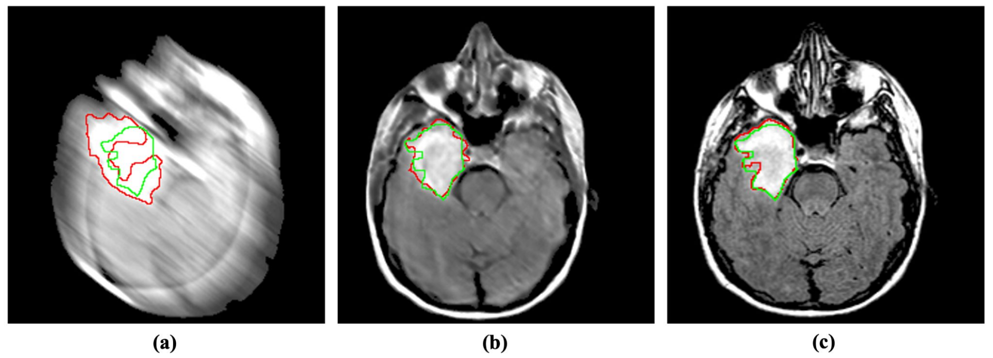
FREDY RAFAEL, MENCIA GONZALEZ, IVAN ANDRES

IMAGE SHARPENING FROM BLURRED IMAGE

- Images are an essential part of communication and information

- A common problem is the loss of quality and sharpness
  - motion, incorrect focus, or limitations of the camera sensor

- Blurred images can compromise effectiveness of critical activities, such as medical imaging, security surveillance, and automatic document interpretation



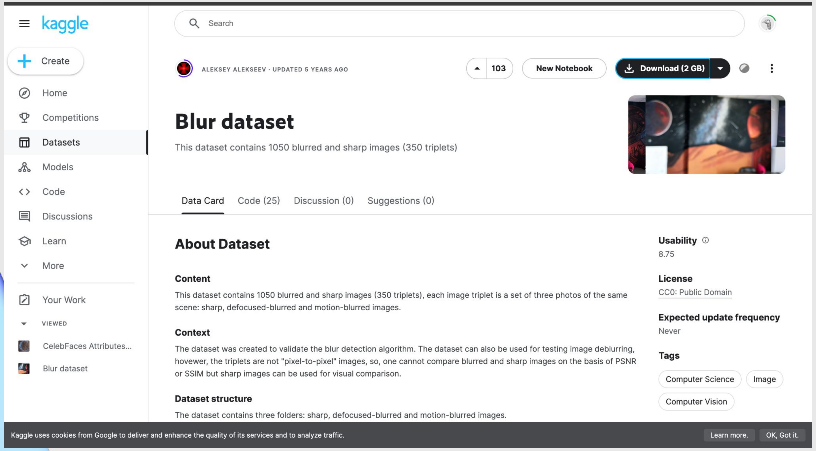
OUR SOLUTION

Develop a machine learning-based solution that enables users to improve the sharpness of blurred images efficiently and automatically through an easy-to-use web interface

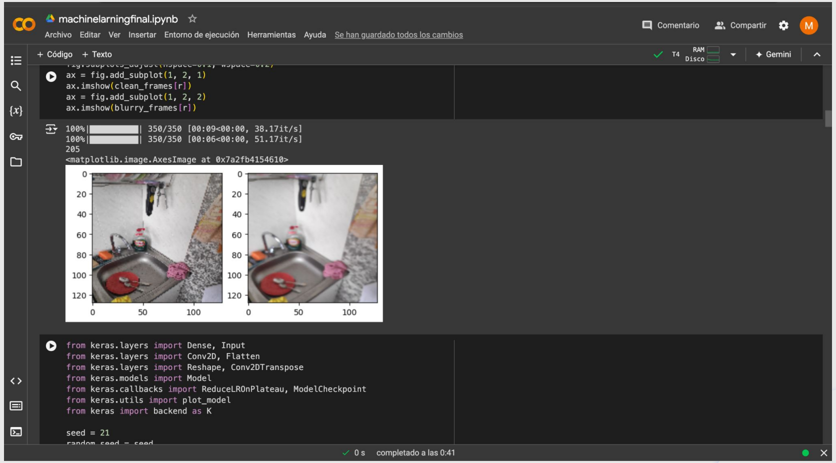
others= simple python code that with the pixels changes the images
ours= machine learning

Technical work
MACHINE LEARNING

THE DATASET WE USED



THESE ARE THE IMAGES



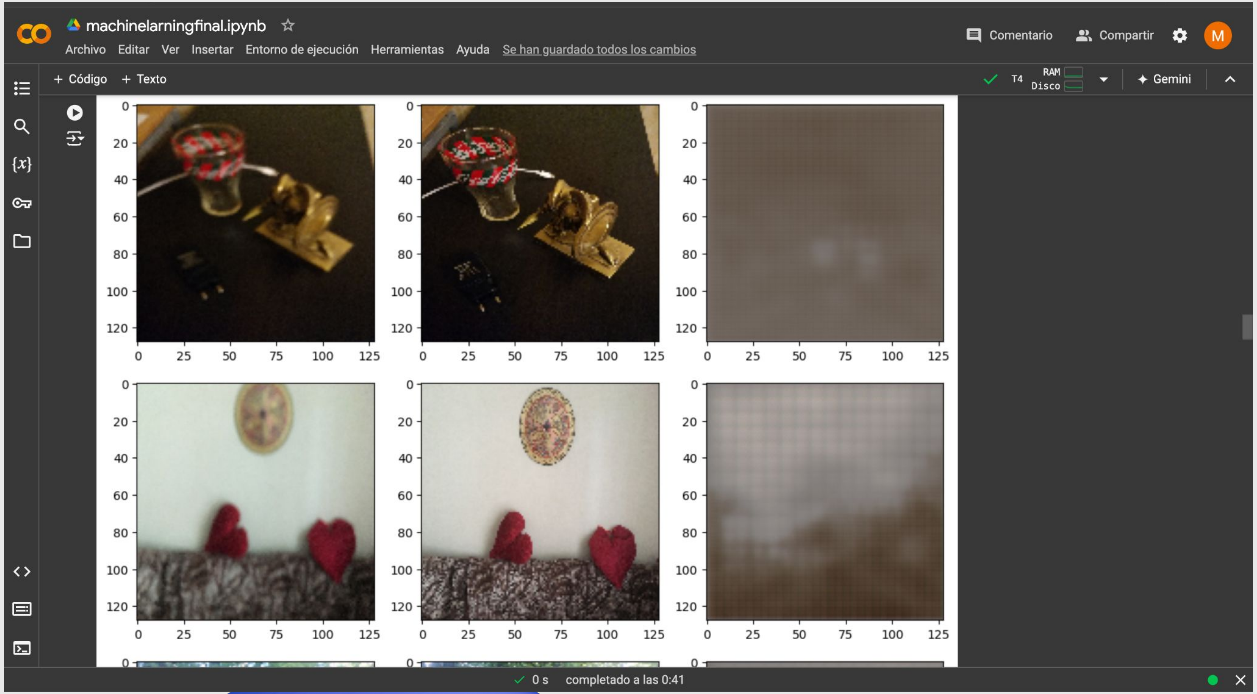
CODE FOR LEARNING WITH 5 IMAGES AS EXAMPLE

machinelarningfinal.ipynb

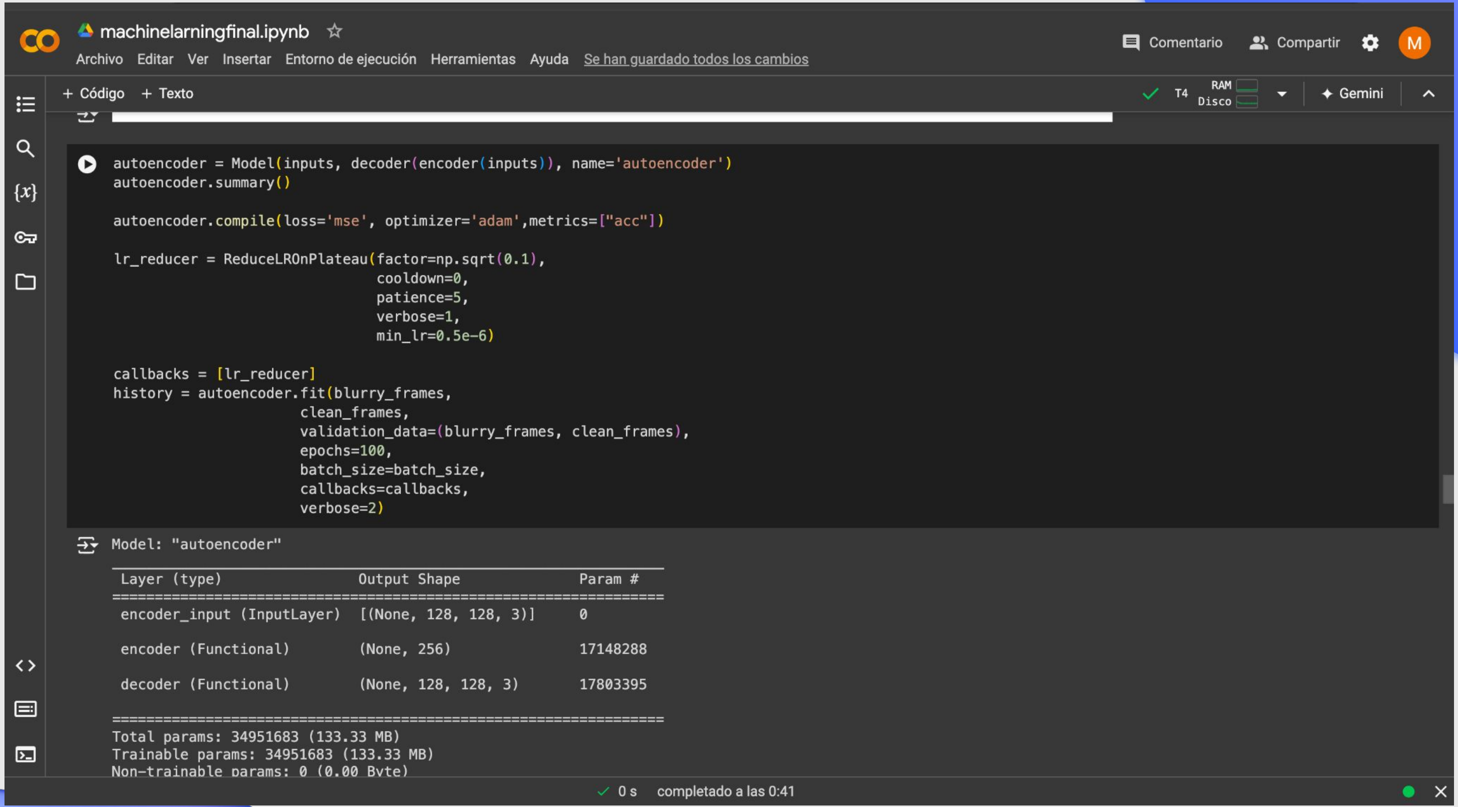
```
autoencoder = Model(inputs, decoder(encoder(inputs)), name='autoencoder')
autoencoder.summary()

autoencoder.compile(loss='mse', optimizer='adam',metrics=["acc"])

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               verbose=1,
                               min_lr=0.5e-6)

callbacks = [lr_reducer]
history = autoencoder.fit(blurry_frames,
                      clean_frames,
                      validation_data=(blurry_frames, clean_frames),
                      epochs=5,
                      batch_size=batch_size,
                      callbacks=callbacks,
                      verbose=2)
```

```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_input (InputLayer)  [(None, 128, 128, 3)]     0

 encoder (Functional)        (None, 256)               17148288

 decoder (Functional)        (None, 128, 128, 3)       17803395

=================================================================
Total params: 34951683 (133.33 MB)
Trainable params: 34951683 (133.33 MB)
```

OUTPUT AFTER LEARNING WITH 5 IMAGES

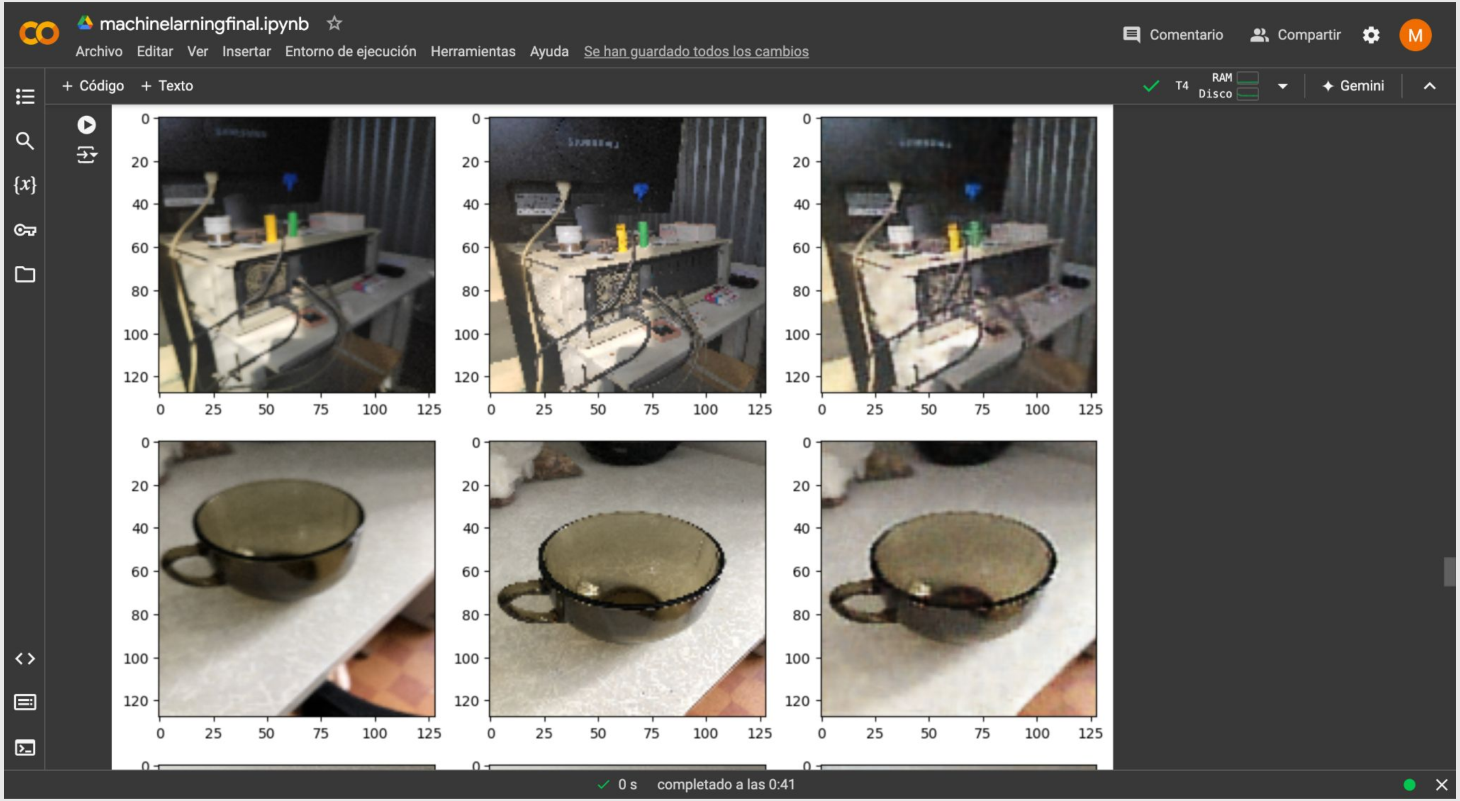


IT WASN'T ENOUGH SO WE MADE IT LEARN WITH 200 IMAGES MORE



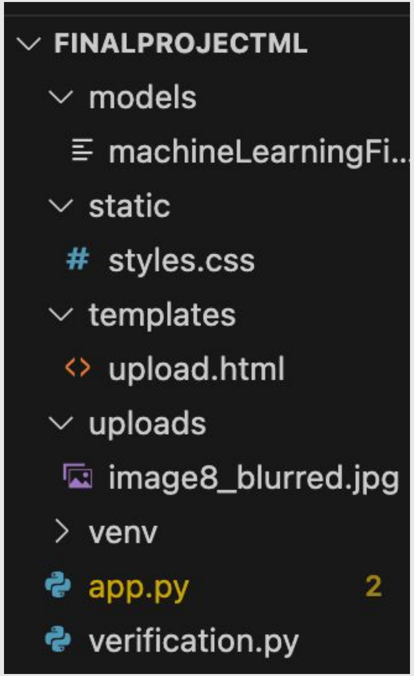

```python
autoencoder = Model(inputs, decoder(encoder(inputs)), name='autoencoder')
autoencoder.summary()

autoencoder.compile(loss='mse', optimizer='adam',metrics=["acc"])

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               verbose=1,
                               min_lr=0.5e-6)

callbacks = [lr_reducer]
history = autoencoder.fit(blurry_frames,
                      clean_frames,
                      validation_data=(blurry_frames, clean_frames),
                      epochs=100,
                      batch_size=batch_size,
                      callbacks=callbacks,
                      verbose=2)
```

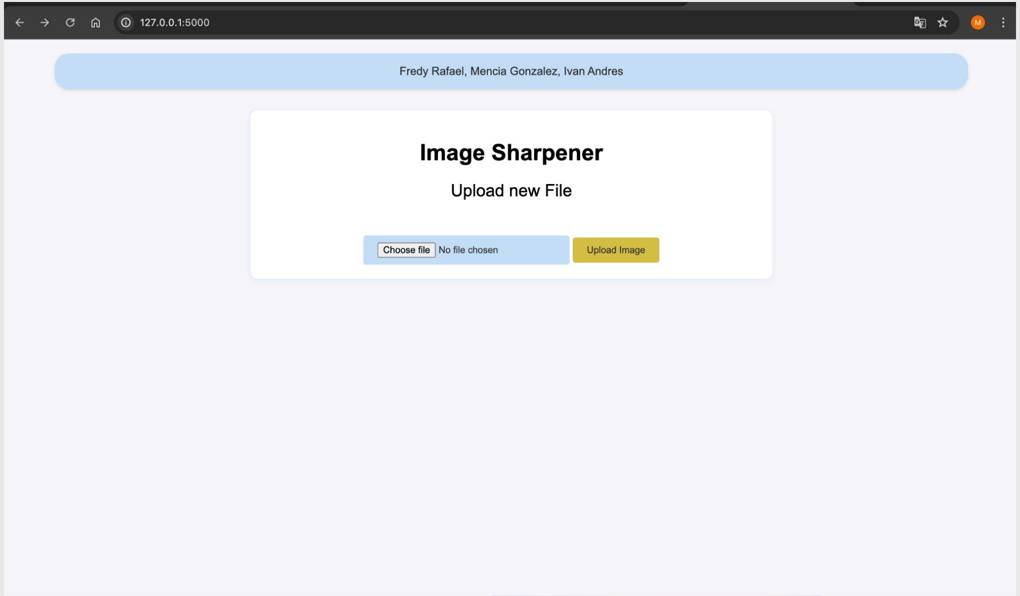
```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_input (InputLayer)  [(None, 128, 128, 3)]     0

 encoder (Functional)        (None, 256)               17148288

 decoder (Functional)        (None, 128, 128, 3)       17803395

=================================================================
Total params: 34951683 (133.33 MB)
Trainable params: 34951683 (133.33 MB)
Non-trainable params: 0 (0.00 Byte)
```


machinelarningfinal.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se han guardado todos los cambios
Comentario
Compartir
M
+ Código + Texto
T4
RAM
Disco
Gemini
0 20 40 60 80 100 120
0 25 50 75 100 125
0 s completado a las 0:41

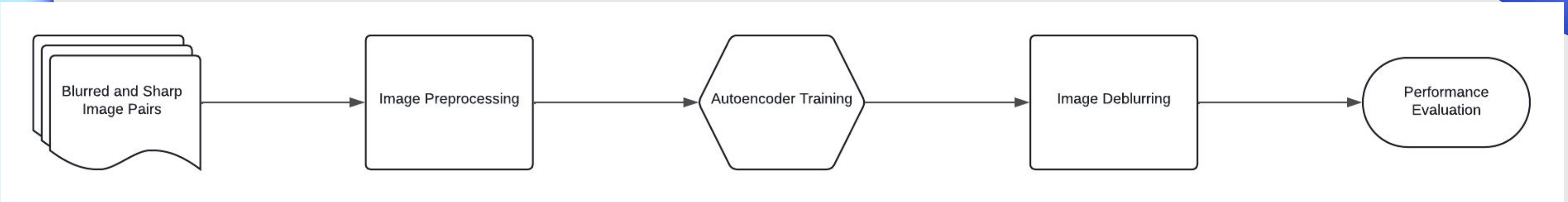
Technical work

WEBSITE



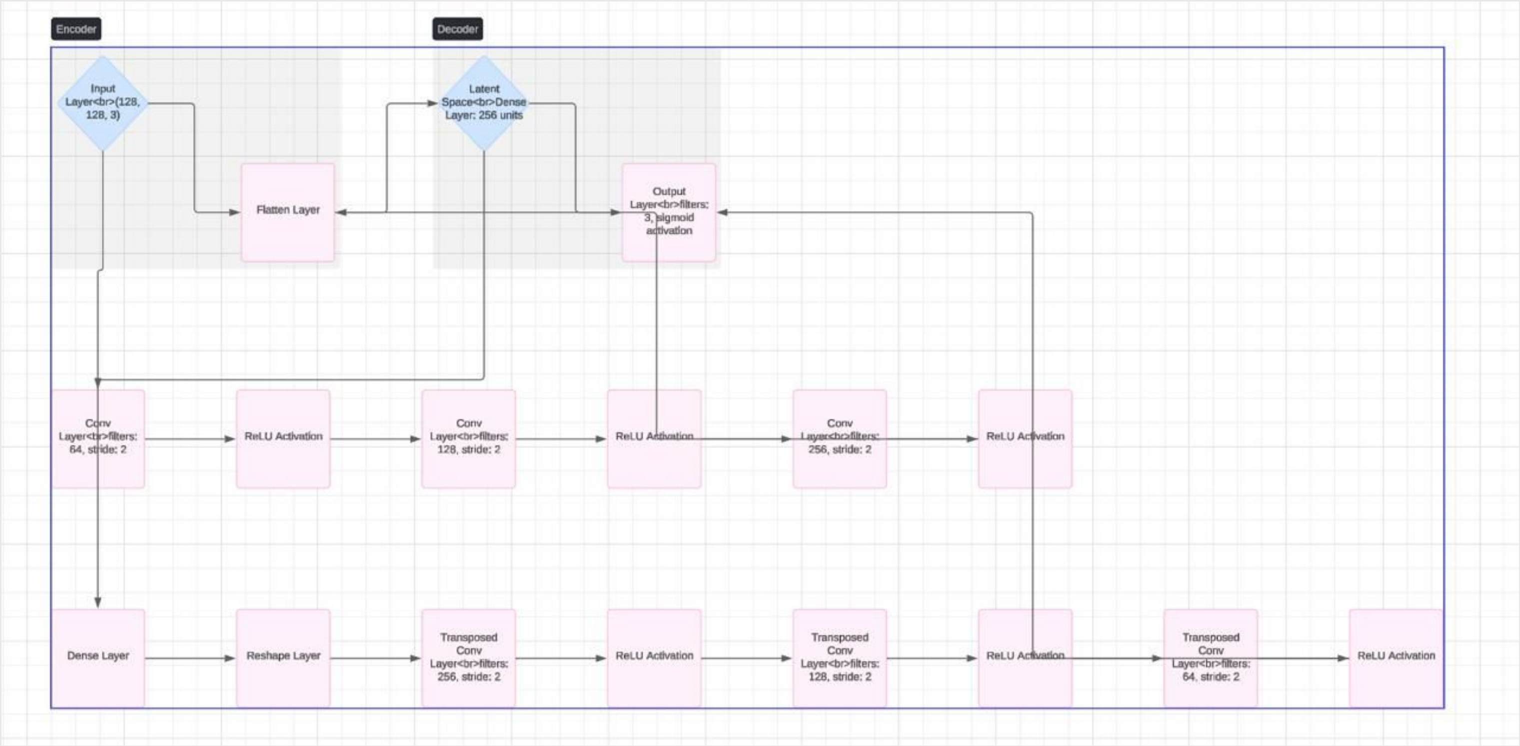


WorkFlow Diagram



Literature review table

Autor	Year	Used Method	Results	comparison with other works	Aplications
Smith et al.	2020	CNN	15% improvement in image clarity	Faster than previous methods, but less accurate than the latest GAN models	Medical diagnostics, security
López y Fernández	2021	GAN	Fine detail recovery in blurred images	Superior in fine details, more computationally expensive	Art restoration, digital media
Zhang et al.	2019	ResNet	High efficiency on low resolution images	Pioneer to use ResNet for this purpose	Surveillance, automotive
Johnson & Kim	2022	MobileNet	Optimization for mobile devices	First energy-efficient mobile fitting	Mobile applications, IoT
Patel et al.	2020	Hybrid CNN-GAN	Combination of CNN and GAN for increased accuracy	Better results than using CNN or GAN separately	Professional photography, graphic design
Nguyen et al.	2021	Autoencoder	Efficient in restoring historical images	Less effective in real time compared to CNN	Historical archiving, museums
Harper and Stone	2023	Q-Learning (Reinforcement Learning)	New approach using reinforcement learning	First exploratory study of its kind	Academic research, experimental development


Encoder
Decoder
Input Layer
(128, 128, 3)
Flatten Layer
Latent Space
Dense Layer: 256 units
Output Layer
filters: 3, sigmoid activation
Conv Layer
filters: 64, stride: 2
ReLU Activation
Conv Layer
filters: 128, stride: 2
ReLU Activation
Conv Layer
filters: 256, stride: 2
ReLU Activation
Dense Layer
Reshape Layer
Transposed Conv Layer
filters: 256, stride: 2
ReLU Activation
Transposed Conv Layer
filters: 128, stride: 2
ReLU Activation
Transposed Conv Layer
filters: 64, stride: 2
ReLU Activation

THANK YOU

!!!