ANEXOS, SISTEMA DE RECONOCIMIENTO FACIAL CON MASCARILLAS

David Alfredo Torres Montalvo

Junior Raúl Sánchez Suárez

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
2022

ANEXOS, SISTEMA DE RECONOCIMIENTO FACIAL CON MASCARILLAS

David Alfredo Torres Montalvo

Junior Raúl Sánchez Suárez

Anexos del libro de trabajo de Grado para optar al título de Ingeniero Electrónico

Director
Jaime Guillermo Barrero Perez
Mg. en potencia eléctrica

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Bucaramanga
2022

Índice general

Índice de figuras

Índice de tablas

1. Metodología

1.1. Modelos con Redes neuronales

Una vez explicada la metodología con un algoritmo que usa Machine learning, se propone un nuevo algoritmo que usa dos modelos de Redes neuronales, una para encontrar la región de interés en la imagen, esta región de interés seria el o los rostro dentro de la imagen y la otra Red neuronal con el objetivo de clasificar si esa región de interés hace parte de un rostro dentro de las tres clases ya planteadas con anterioridad, sin mascarilla, mascarilla mal puesta, y con mascarilla. para ilustra esta metodología se expone un diagrama que contiene el procedimiento antes descrito a groso modo.

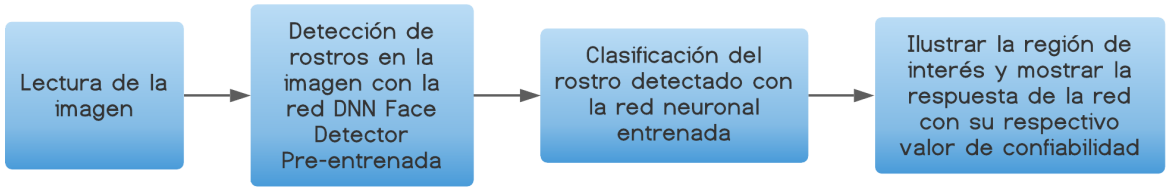

Figura 1.1: Procesamiento y clasificación del algoritmo.

1.1.1. Preprocesado de las imágenes

A diferencia de los algoritmos que hacen uso de Machine learning, para el algoritmo con redes neuronales solo fue necesario encontrar el recorte del rostro dentro la imagen, no hubo la necesidad de encontrar puntos claves del rostro como lo es la posición de los ojos para poder aplicar una rotación y tener una imagen mas parecida en ángulo y postura con respecto a las que fueron usadas en el entrenamiento de los modelos LBPH y Eigenface.

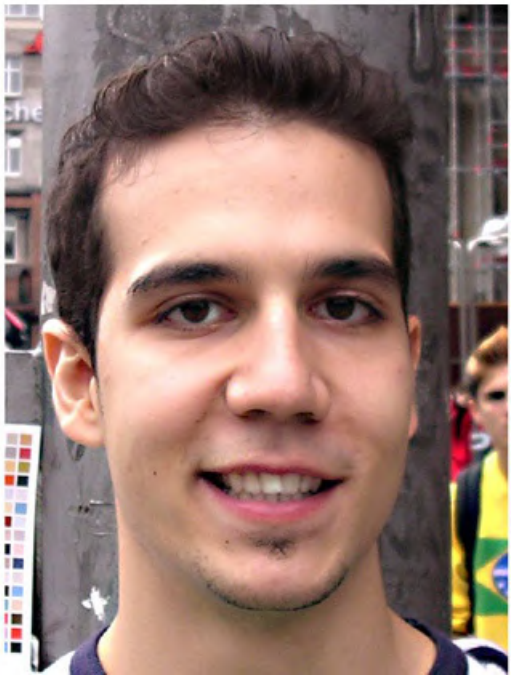

Figura 1.2: Imagen de entrada en la red pre-entrenada DNN Face Detector

Figura 1.3: Recorte del rostro dentro de la imagen a la salida de la red

1.1.2. Entrenamiento de los modelos

Al igual que en el entrenamiento de los modelos de machine learning LBPH y Eigenface, en el script que hace el entrenamiento de los modelos, lo primero que se hace es leer de

forma iterativa y ordenada las imágenes de la base de datos, que están de nuevo separada en tres carpetas distintas con el objetivo de facilitar el etiquetado, estas imágenes y etiquetas se van almacenando en listas cuya posición de imagen en la primera lista, corresponde a una etiqueta de caracteres string en la segunda lista, contiguamente se debe de hacer una binarización de las etiquetas para poder así introducirlas en la función de entrenamiento de la red, en dicha función se especifica a voluntad que el 20 % de las imágenes del dataset fueron usadas para examinar la red en el proceso de entrenamiento.

Los parámetros de configuración usados para generar imágenes de entrenamiento por data augmentation son los siguientes:

- rotation_range=20
- zoom_range=0.15
- width_shift_range=0.2
- height_shift_range=0.2
- shear_range=0.15
- horizontal_flip=True
- fill_mode="nearest")

otro aspecto destacable es que se usaron los pesos de la red MObileNetV2 para el modelo base y por ultimo las imágenes de entrenamiento fueron escaladas a una resolución de 224 x 224 píxeles

El código del entrenamiento de los modelos se puede encontrar en (Torres y Sánchez, 2022) , siguiendo la ruta /Facemask-detection-codes /Network codes /Train.ipynb

1.1.3. Parámetros de rendimiento en el entrenamiento

Al momento de entrenar los modelos que en este caso son cuatro, se consiguió abstraer parámetros de rendimiento en el proceso, entre estos están el desempeño en precisión y error de las épocas que fueron seleccionadas (20 Épocas), las funciones de perdida y ganancia a medida que se va entrenado y por ultimo la precisión con el porcentaje del dataset designado a un test de prueba. Para cada uno de estos modelos la cantidad de imágenes con las que fueron entrenados van desde los 1000 a 2000 imágenes por clase con incrementos de imágenes de 500 en 500 a excepción del ultimo modelo que fue entrenado con todo el dataset, que suma una cantidad de aproximadamente un poco menos de 2400 imágenes por clase, cabe especificar que también se uso el 20 % de las imágenes del dataset como prueba en el momento de entrenar así que el contenido neto del conjunto de datos de entrenamiento es realmente menor al mencionado.

1.1.3.1. Entrenamiento Modelo V1

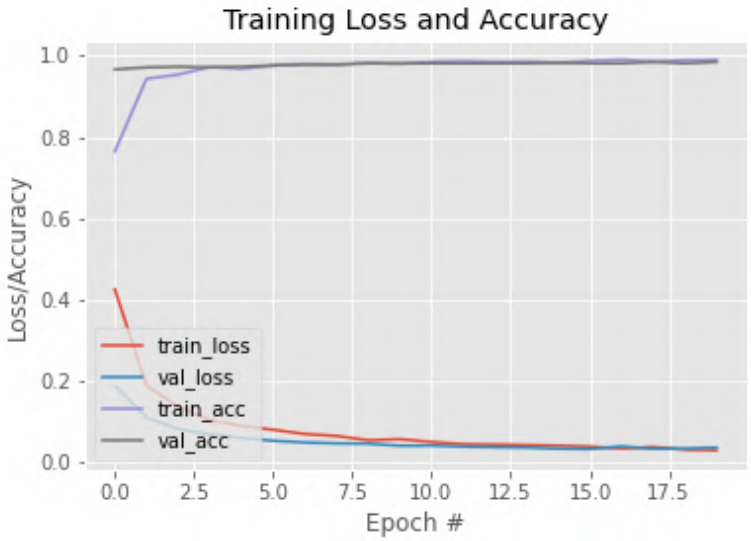

Figura 1.4: Modelo V1: Función de perdidas y ganancias

```
[INFO] evaluating network...
              precision    recall  f1-score   support

  error_mask       0.99      0.99      0.99       200
   with_mask       0.97      0.99      0.98       200
without_mask       0.99      0.97      0.98       200

    accuracy                           0.98       600
   macro avg       0.99      0.98      0.98       600
weighted avg       0.99      0.98      0.98       600
```

Figura 1.5: Modelo V1: Test del modelo con el 20 % del dataset

1.1.3.2. Entrenamiento Modelo V2

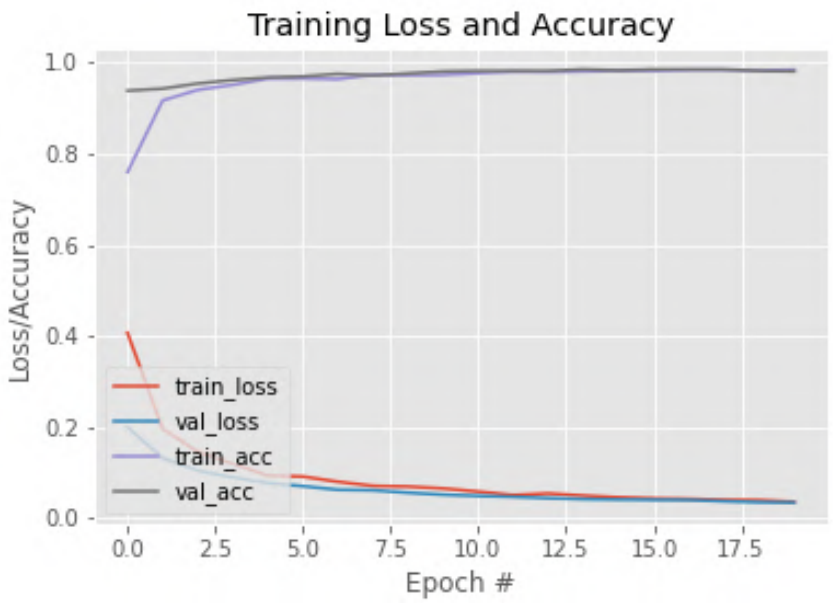

Figura 1.6: Modelo V2: Función de perdidas y ganancias

```
[INFO] evaluating network...
              precision    recall  f1-score   support

  error_mask       0.97      0.98      0.98       300
   with_mask       0.98      0.97      0.97       300
without_mask       0.99      0.99      0.99       300

    accuracy                           0.98       900
   macro avg       0.98      0.98      0.98       900
weighted avg       0.98      0.98      0.98       900
```

Figura 1.7: Modelo V2: Test del modelo con el 20 % del dataset

1.1.3.3. Entrenamiento Modelo V3

Figura 1.8: Modelo V3: Función de perdidas y ganancias

	precision	recall	f1-score	support
error_mask	0.99	0.98	0.99	400
with_mask	0.98	0.99	0.98	400
without_mask	1.00	0.99	0.99	400
accuracy			0.99	1200
macro avg	0.99	0.99	0.99	1200
weighted avg	0.99	0.99	0.99	1200

Figura 1.9: Modelo V3: Test del modelo con el 20 % del dataset

1.1.3.4. Entrenamiento Modelo V4

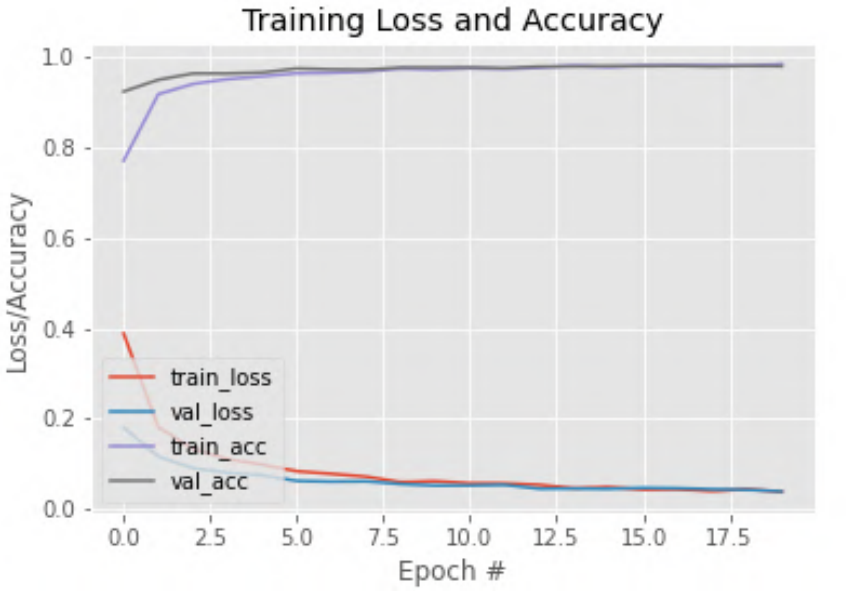

Figura 1.10: Modelo V4: Función de perdidas y ganancias

```
[INFO] evaluating network...
              precision    recall  f1-score   support

  error_mask       0.99      0.98      0.98       479
   with_mask       0.96      0.98      0.97       475
without_mask       0.99      0.98      0.99       477

    accuracy                           0.98      1431
   macro avg       0.98      0.98      0.98      1431
weighted avg       0.98      0.98      0.98      1431
```

Figura 1.11: Modelo V4: Test del modelo con el 20 % del dataset

1.1.4. Código del algoritmo

De nuevo el algoritmos se ajustó para poder realizar el reconocimiento de mascarillas en distintos tipos de contenido multimedia que hagan uso del aspecto visual, entre estos encontramos imágenes, vídeos y por ultimo video stream en la plataforma en la cual quiera ser adaptado y ejecutado el código, como se ha mencionado antes, el algoritmo consta de una

primera parte que es la de reconocer los rostros en las imágenes con la red neuronal DNN Face Detector que hace uso de los pesos de la red Caffemodel después de haber seleccionado la o las regiones de interés dentro de esta imagen o frame, hace el recorte del rostro y lo envía a clasificar en el modelo entrenado, por ultimo mediante unas pequeñas lineas de código final, se escoge la confiabilidad mas alta que arroja en respuesta el modelo y con ello se procede a en marcar el rostro en diferentes colores junto a la impresión del valor del porcentaje de con fiabilidad y la etiqueta de las tres clases escogidas. El resultado de previa descripción se muestra en las figuras subsecuentes:

Nuevamente el código del algoritmo usado en el reconocimiento facial de mascarillas con redes neuronales se puede encontrar en (Torres y Sánchez, 2022), en la ruta /Facemask-detection-codes /Network codes/detect_mask_images_full.py

Estos algoritmos están basados en los encontrados por la búsqueda bibliográfica (Comunidad AIRX, s.f.).

Figura 1.12: Clasificación de: con mascarilla y confiabilidad del 100 %

Figura 1.13: Clasificación de: sin mascarilla y confiabilidad del 100 %

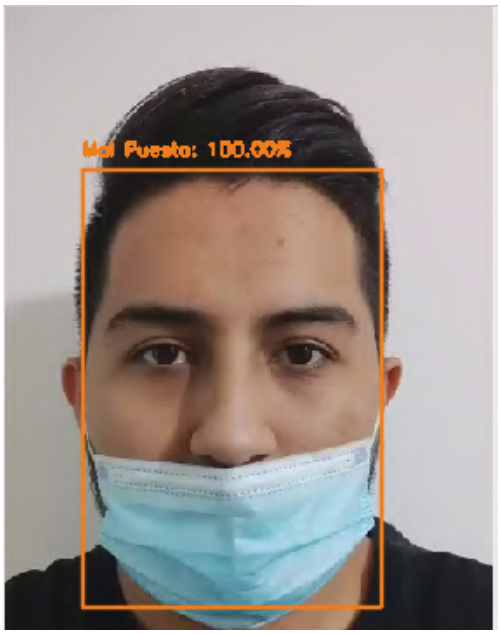

Figura 1.14: Clasificación de: mascarilla mal puesta y confiabilidad del 100 %

2. Resultados

Para poder evaluar los resultados obtenidos tanto con los modelos que hacen uso de Machine learning como de Redes neuronales, se emplean y se comparan diferentes términos y parámetros de rendimiento tales como: la precisión, el error, tiempo de clasificación y el tiempo de ejecución de los diversos algoritmos que hacen uso de diferentes clasificadores entrenados de forma independiente con cierta cantidad en ascenso de imágenes.

2.1. Resultados con Machine learning

Como se había mencionado en el documento principal, los modelos de clasificación usados con Machine learning son : Eigenface y Local Binary Pattern Histogram. La forma en la que se obtuvieron los distintos parámetros de rendimiento acontece de la organización previa de imágenes, etiquetadas mediante una revisión visual por parte de los estudiantes desarrolladores del proyecto, en la cual se clasificaron 150 imágenes por clase, dando como resultado 450 imágenes examinadas por parte de cada modelo. Es importante mencionar que la cantidad de modelos de Eigenface y LBPH que se evaluaron suman en total 10 modelos diferentes, 5 modelos Eigenface y 5 modelos LBPH. Para cada uno de estos modelos la cantidad de imágenes con las que fueron entrenados van desde las 800 a 1200 imágenes por clase con incrementos de imágenes de 100 en 100. Se aclara a manera de ejemplo que las imágenes de la base de datos de entrenamiento usadas para obtener un modelo de 1100 imágenes por clase son las mismas tanto para Eigenface como para LBPH. En las siguientes tablas i/o figuras se representa los parámetros de rendimiento para cada clase de clasificación con una base de datos de prueba con la cantidad de imágenes por clase antes mencionada.

2.1.1. Plataforma: PC

La computadora portátil que ejecuto el algoritmo en lenguaje Python y produjo los resultados adjuntos tiene las siguientes especificaciones:

- Procesador: Intel Core I5 8250U
- Ram: 12 GB DDR4 a 2400 MHz
- SSD 500 GB SATA3 M.2
- SO: Ubuntu 20.04 LTS

2.1.1.1. Rendimiento en: sin mascarilla

Sin mascarilla					
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
Eigenface 800	98.63	1.36	15.95	31.30	466
Eigenface 900	98.63	1.36	17.72	31.30	545.4
Eigenface 1000	99.32	0.68	20.32	33.91	629.6
Eigenface 1100	99.32	0.68	23.81	38.04	718.5
Eigenface 1200	99.32	0.68	26.34	39.92	812.2
LBPH 800	98.63	1.36	140.50	152.84	188.5
LBPH 900	99.32	0.68	156.94	169.10	212
LBPH 1000	99.32	0.68	175.17	187.77	235.6
LBPH 1100	98.63	1.36	192.72	204.90	259.3
LBPH 1200	98.63	1.31	210.17	222.62	282.9

Figura 2.1: Tabla de parámetros de rendimiento para clase de: sin mascarilla

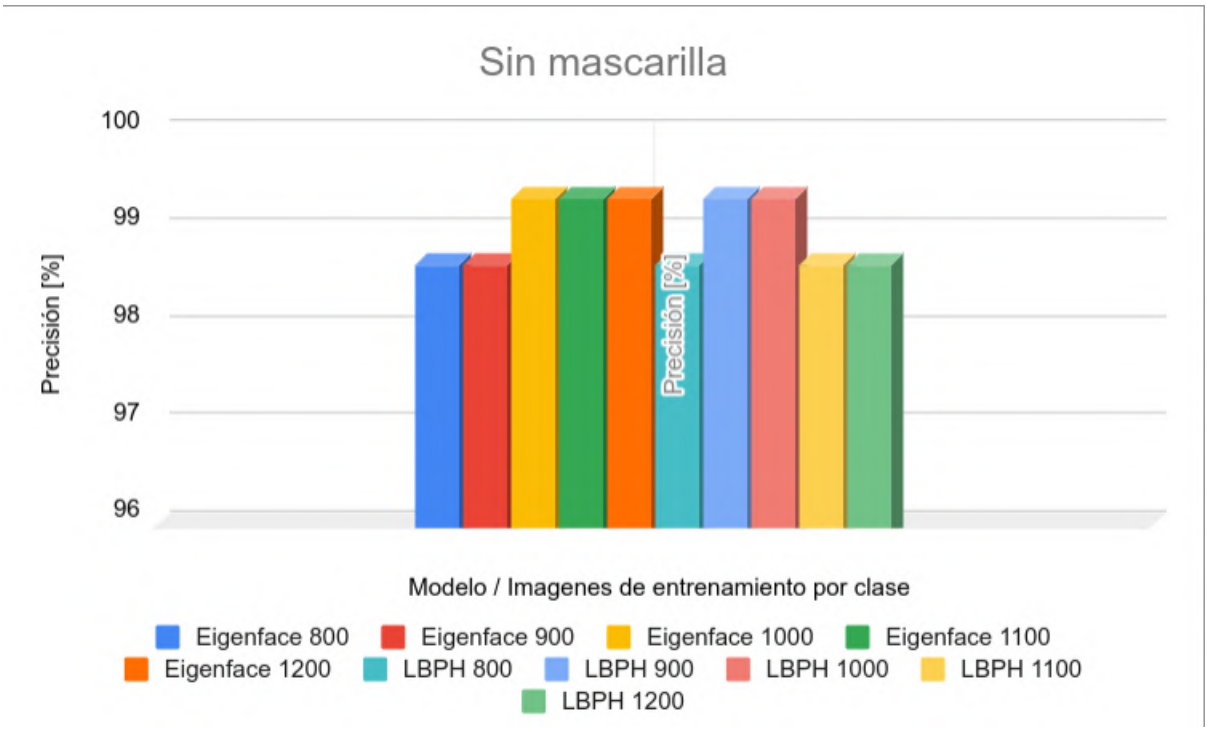

Figura 2.2: Gráfica de precisión para clase de: sin mascarilla

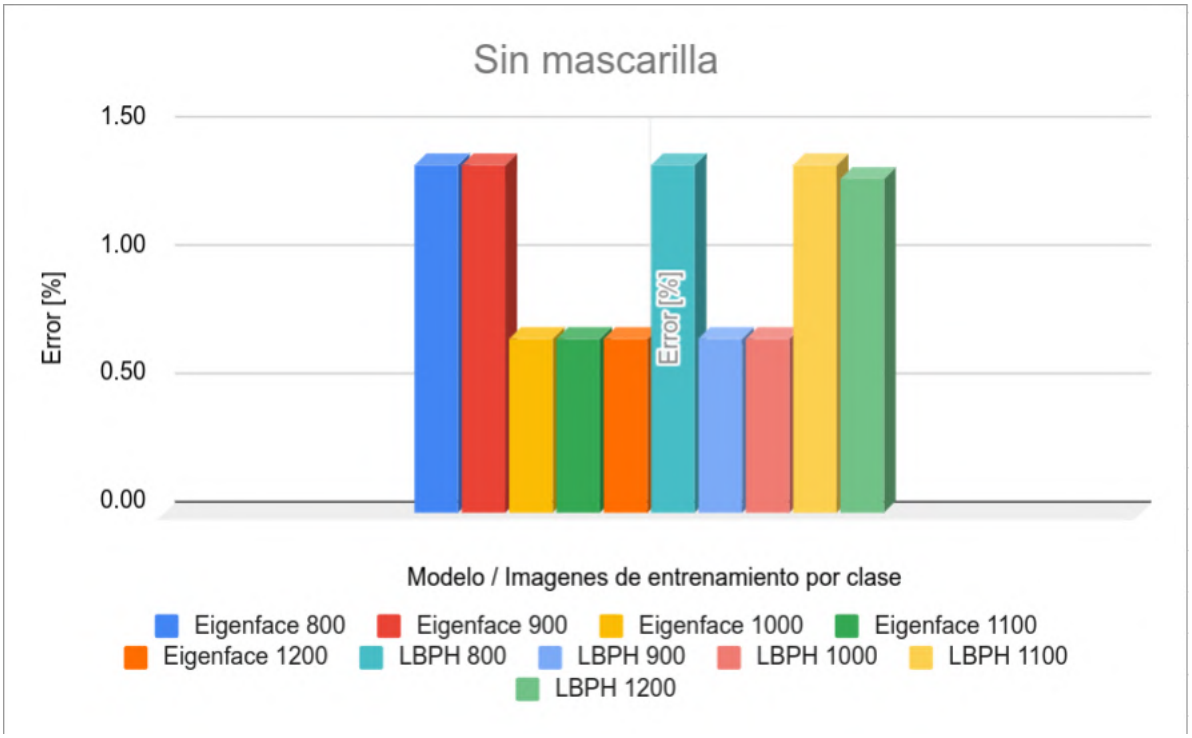

Figura 2.3: Gráfica de Error para clase de: sin mascarilla

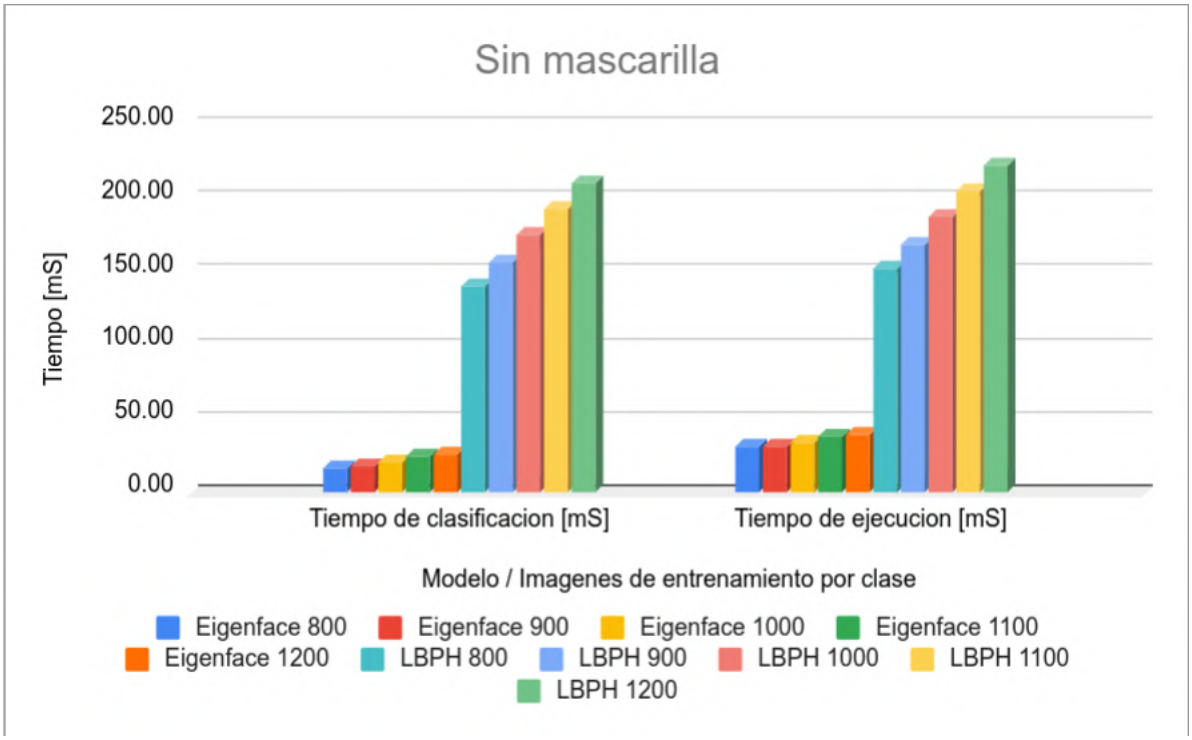

Figura 2.4: Gráfica de tiempos para clase de: sin mascarilla

2.1.1.2. Rendimiento en: mascarilla mal puesta

Error mascarilla					
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
Eigenface 800	93.28	6.71	15.13	36.26	466
Eigenface 900	93.28	6.71	17.29	37.70	545.4
Eigenface 1000	91.94	8.05	20.72	41.86	629.6
Eigenface 1100	94.63	5.36	24.13	47.34	718.5
Eigenface 1200	94.63	5.36	26.66	47.68	812.2
LBPH 800	94.63	5.36	139.70	158.06	188.5
LBPH 900	93.96	6.04	156.72	175.17	212
LBPH 1000	93.96	6.04	173.51	191.56	235.6
LBPH 1100	93.29	6.71	191.20	209.29	259.3
LBPH 1200	93.96	6.04	209.48	227.92	282.9

Figura 2.5: Tabla de parámetros de rendimiento para clase de: mascarilla mal puesta

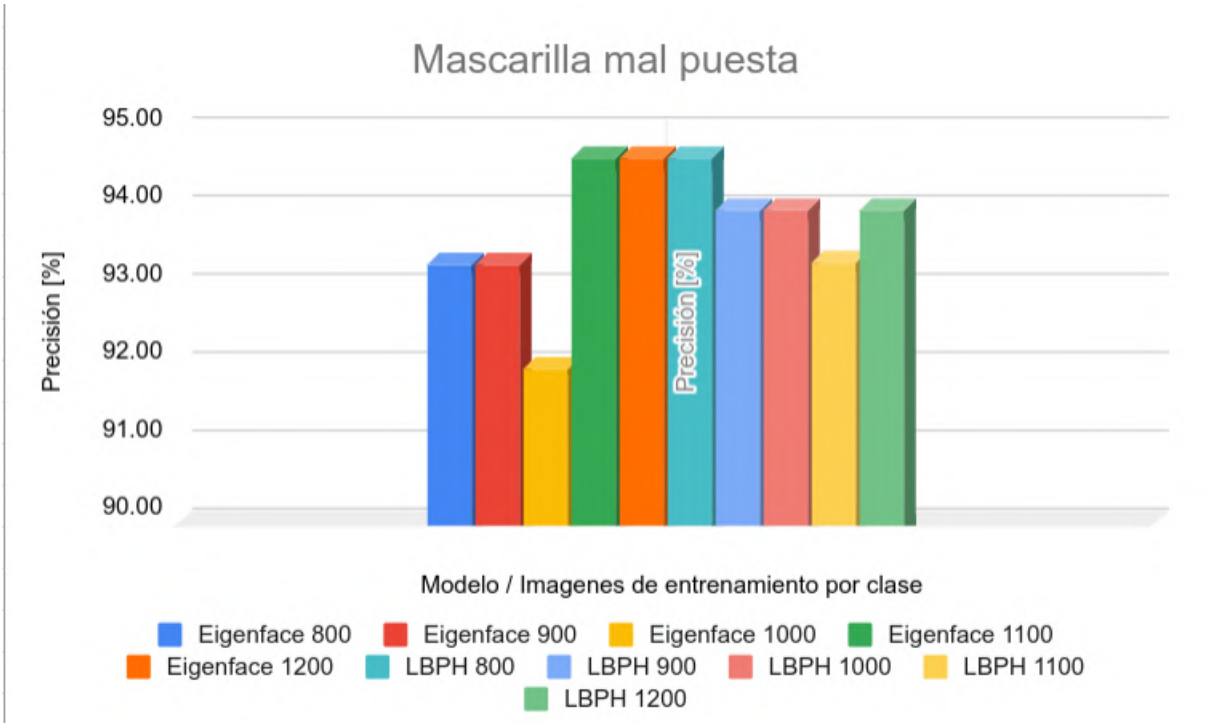

Figura 2.6: Gráfica de precisión para clase de: mascarilla mal puesta

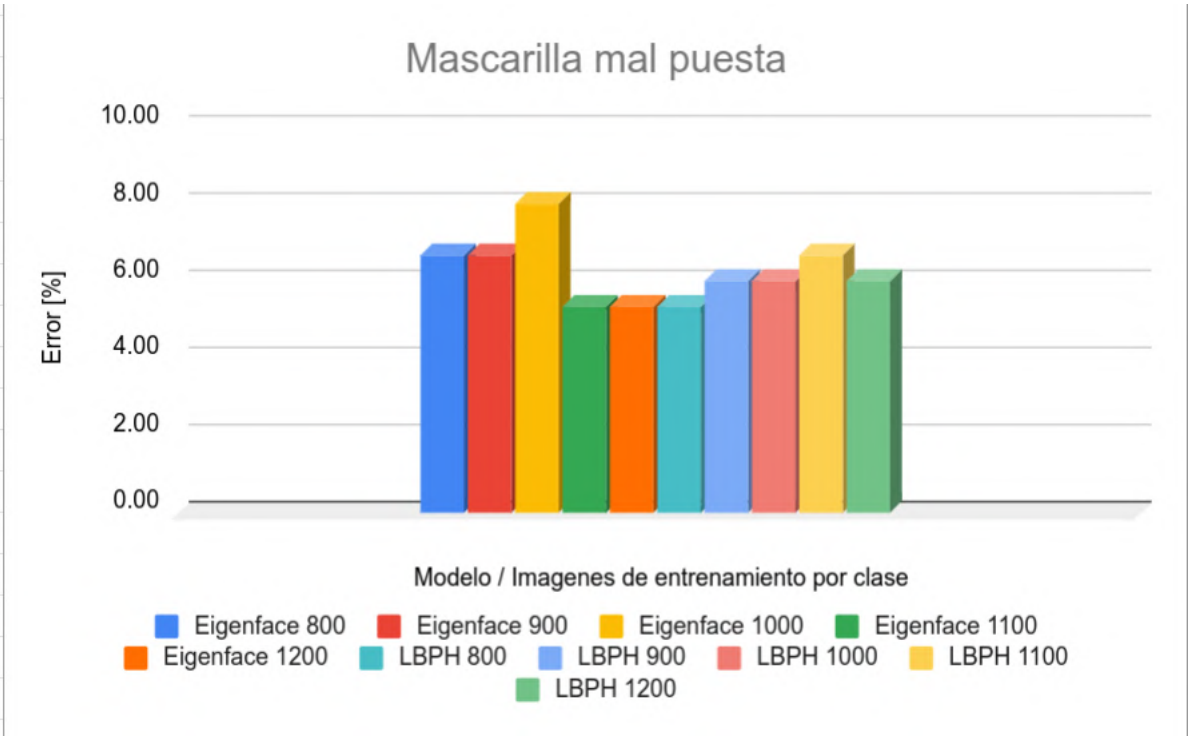

Figura 2.7: Gráfica de Error para clase de: mascarilla mal puesta

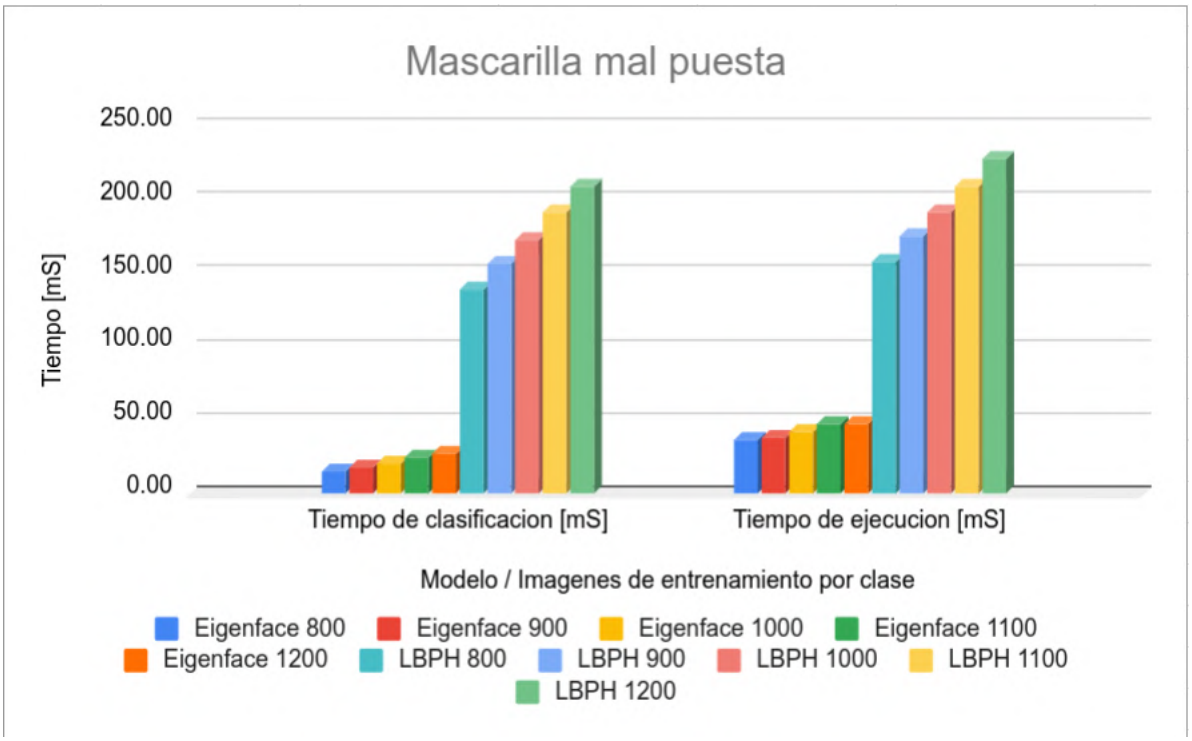

Figura 2.8: Gráfica de tiempos para clase de: mascarilla mal puesta

2.1.1.3. Rendimiento en: con mascarilla

Con mascarilla					
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
Eigenface 800	93.10	6.89	14.43	40.16	466
Eigenface 900	91.03	8.96	16.95	42.77	545.4
Eigenface 1000	90.35	9.65	19.78	45.19	629.6
Eigenface 1100	90.35	9.65	22.35	47.87	718.5
Eigenface 1200	91.72	8.27	25.66	51.21	812.2
LBPH 800	96.55	3.44	143.64	166.95	188.5
LBPH 900	97.24	2.75	161.26	184.47	212
LBPH 1000	97.24	2.75	179.34	202.85	235.6
LBPH 1100	97.24	2.75	197.30	220.25	259.3
LBPH 1200	97.93	2.06	214.77	237.68	282.9

Figura 2.9: Tabla de parámetros de rendimiento para clase de: con mascarilla

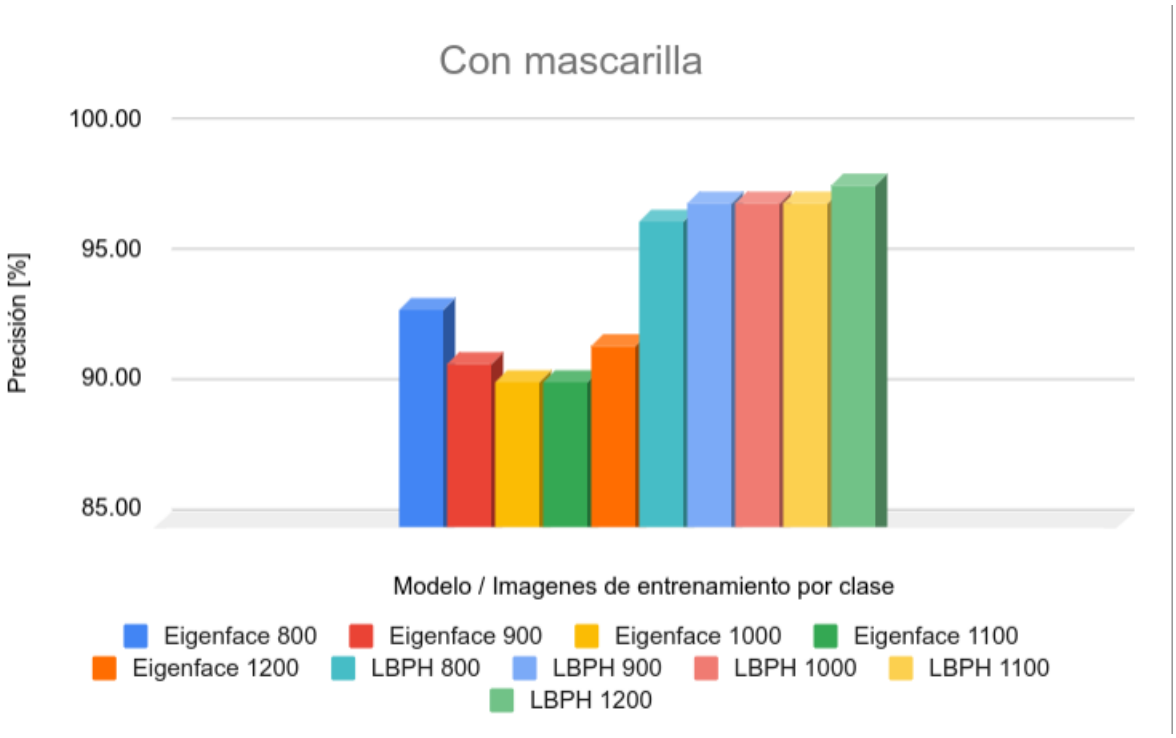

Figura 2.10: Gráfica de precisión para clase de: con mascarilla

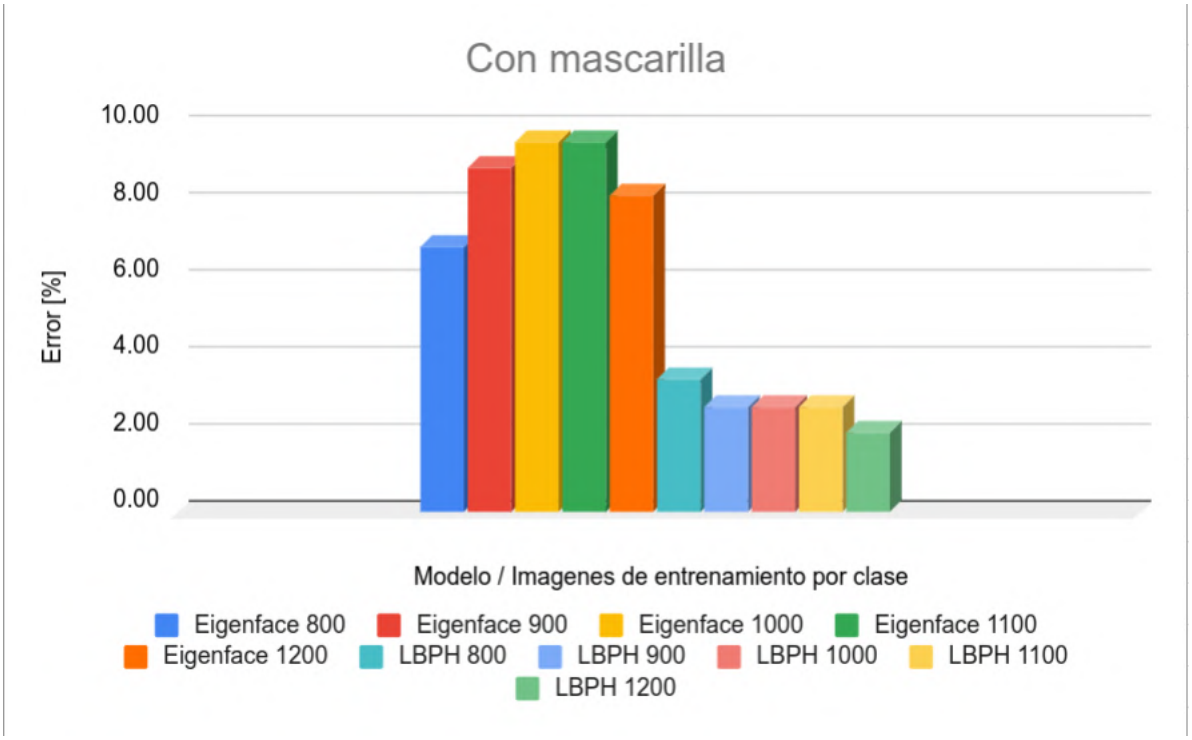

Figura 2.11: Gráfica de Error para clase de: con mascarilla

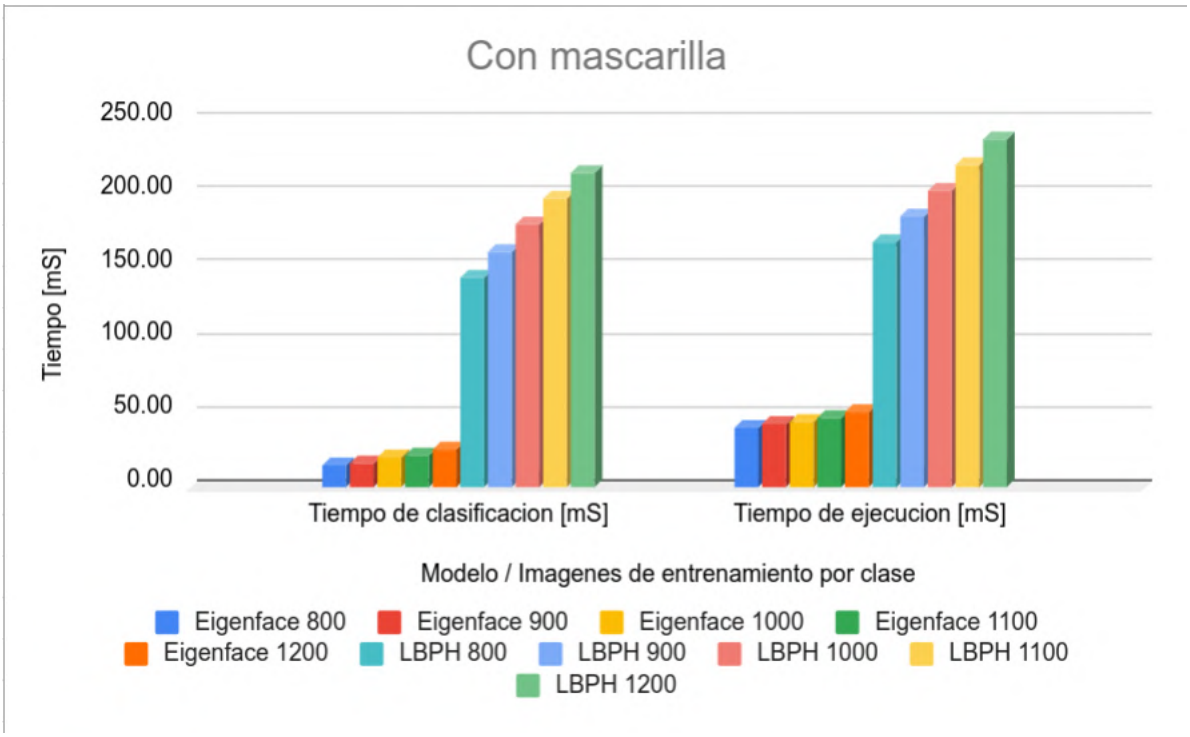

Figura 2.12: Gráfica de tiempos para clase de: con mascarilla

2.1.1.4. Matrices de confusión en PC.

Matriz de Confusión Modelo LBPH 800 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	140	0	1
	Mascarilla erronea	5	141	1
	Sin mascarilla	0	8	145

Tabla 2.1: Matriz de confusión para modelo LBPH de 800 imágenes

Matriz de Confusión Modelo LBPH 900 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	141	0	0
	Mascarilla erronea	4	140	1
	Sin mascarilla	0	9	146

Tabla 2.2: Matriz de confusión para modelo LBPH de 900 imágenes

Matriz de Confusión Modelo LBPH 1000 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	141	0	0
	Mascarilla erronea	4	140	1
	Sin mascarilla	0	9	146

Tabla 2.3: Matriz de confusión para modelo LBPH de 1000 imágenes

Matriz de Confusión Modelo LBPH 1100 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	141	1	0
	Mascarilla erronea	4	139	2
	Sin mascarilla	0	9	145

Tabla 2.4: Matriz de confusión para modelo LBPH de 1100 imágenes

Matriz de Confusión Modelo LBPH 1200 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	142	0	0
	Mascarilla erronea	3	140	2
	Sin mascarilla	0	9	145

Tabla 2.5: Matriz de confusión para modelo LBPH de 1200 imágenes

Matriz de Confusión Modelo EigenFace 800 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	135	7	2
	Mascarilla erronea	9	139	0
	Sin mascarilla	1	3	145

Tabla 2.6: Matriz de confusión para modelo EigenFace de 800 imágenes

Matriz de Confusión Modelo EigenFace 900 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	132	7	2
	Mascarilla erronea	11	139	0
	Sin mascarilla	2	3	145

Tabla 2.7: Matriz de confusión para modelo EigenFace de 900 imágenes

Matriz de Confusión Modelo EigenFace 1000 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	131	6	1
	Mascarilla erronea	9	137	0
	Sin mascarilla	5	6	146

Tabla 2.8: Matriz de confusión para modelo EigenFace de 1000 imágenes

Matriz de Confusión Modelo EigenFace 1100 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	131	3	1
	Mascarilla erronea	9	141	0
	Sin mascarilla	5	5	146

Tabla 2.9: Matriz de confusión para modelo EigenFace de 1100 imágenes

Matriz de Confusión Modelo EigenFace 1200 imagenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	133	5	1
	Mascarilla erronea	7	141	0
	Sin mascarilla	5	3	146

Tabla 2.10: Matriz de confusión para modelo EigenFace de 1200 imágenes

Las gráficas i/o tablas anteriormente mostradas corresponden a una precisión y error que proviene de una suma de etiquetas arrojadas por el modelo en cuestión, por supuesto dividiendo sobre el numero total de imágenes evaluadas por clase. Para obtener los datos de tiempo, la primera bandera que se usa es para el tiempo de clasificación y se toma justo antes de ejecutar la función de predicción del modelo correspondiente. Por otra parte, la segunda bandera se toma una vez el algoritmo obtiene una respuesta por parte del modelo. En el caso del tiempo de ejecución total, se toma la primera bandera justo antes de leer el frame y la segunda bandera se toma, al igual que en el caso anterior, justo después de obtener una respuesta por el clasificador, de esta forma se han calculado los anteriores parámetros de rendimiento.

Para la parte del análisis de resultados podemos observar en la tabla de la figura 2.1, que a medida que se usan mas imágenes de entrenamiento para el modelo de Eigenface en la clase de sin mascarilla, aumenta la precisión y disminuye el error, no podemos decir lo mismo en el caso de LBPH pues en estos modelos se podría estar presentando problemas de overfitting ya que con pruebas propias se ha analizado previamente que este modelo funciona de forma correcta con unas 800 a 900 imágenes por clase, en ambos modelos se tiene precisiones considerablemente altas, mas del 95 % de las imágenes de la clase en la base de datos fueron clasificadas correctamente. Para la parte de los tiempos de ejecución podemos observar en la tabla de la figura 2.1 y la gráfica de la figura 2.4, que el preprocesado que se le hace a la imagen consume aproximadamente unos 15 mS a 20

mS del tiempo total de ejecución, es notable que el tiempo de clasificación en imágenes es considerablemente menor en el modelo Eigenface a comparación de LBPH.

Tomando como referencia otra clase como los es mascarilla mal puesta, los modelos en este caso, les cuesta mas clasificar correctamente las imágenes, según la tabla de la figura 2.5, la precisión para esta clase ya no supera el 95 % pero de igual forma que en lo descrito anteriormente para los modelos Eigenface a medida que aumenta el numero de imágenes con las cuales fue entrenado dicho modelo aumenta la precisión aunque para el modelo que fue entrenado con 1000 imágenes por clase esta precisión disminuye lo mismo ocurre con el modelo LBPH a medida que aumenta el numero de imágenes de entrenamiento la precisión decae un poco y aumenta en el ultimo modelo de 1100 LBPH.

Por ultimo, el modelo que sale mejor plantado es el LBPH cuando la clase es con mascarilla, si bien Eigenface disminuye su precisión a medida que se entrena con mas imágenes, sigue teniendo una precisión por encima del 90 % en cambio LBPH aumenta su precisión progresivamente y mantiene en el umbral del 96 %

Aspectos comparables entre estos dos modelos es que si bien LBPH es hasta 10 veces mas lento que Eigenface, en imágenes, LBPH es bastante mas preciso y estable al momento de clasificar, de acuerdo al marco teórico, como el proceso de ejecución depende de parámetros como la textura de la imagen, al realizar el histograma, este modelo es mas admisible a los cambios de intensidad de la imagen en escala de grises volviéndolo mas eficaz al momento de clasificar imágenes de rostros que difieran de las usadas en el entrenamiento del modelo.

2.2. Resultados con Redes neuronales

Nuevamente la forma en la que se obtuvieron los distintos parámetros de rendimiento proviene de la organización previa de imágenes, etiquetadas mediante una revisión visual por parte de los estudiantes desarrolladores del proyecto, en la cual se clasificaron 150 imágenes por clase un poco diferentes a las usadas en las pruebas hechas para los modelos de ma-

chine learning, es importante mencionar que la cantidad de modelos examinados con redes neuronales suman una cantidad de cuatro.

2.2.1. Plataforma: PC

La computadora portátil que ejecuto el algoritmo en lenguaje Python y produjo los resultados adjuntos tiene las siguientes especificaciones:

- Procesador: Intel Core I5 8250U
- Ram: 12 GB DDR4 a 2400 MHz
- SSD 500 GB SATA3 M.2
- SO: Ubuntu 20.04 LTS

2.2.1.1. Rendimiento en: sin mascarilla

Sin mascarilla						
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Confiabilidad [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
V1/ 1000	97.97	94.92	2.02	44.45	65.86	11.5
V2 / 1500	98.64	99.61	1.35	44.09	66.06	11.5
V3 / 2000	99.32	99.22	0.68	48.76	72.57	11.5
V4 / ALL	100.00	99.28	0.00	47.35	70.31	11.5

Figura 2.13: Tabla de parámetros de rendimiento para clase de: sin mascarilla

Figura 2.14: Gráfica de precisión para clase de: sin mascarilla

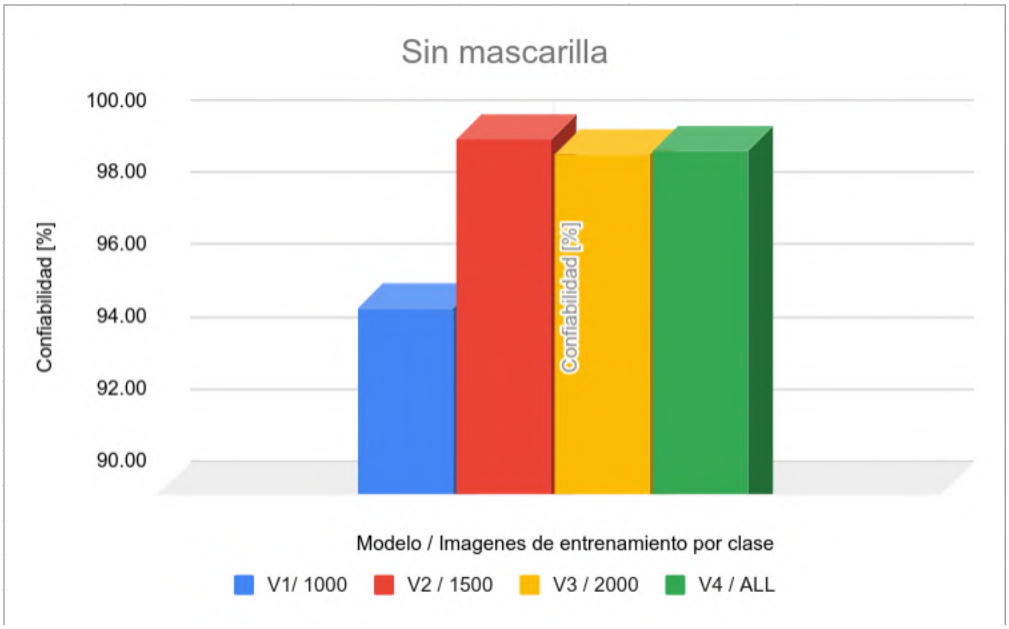

Figura 2.15: Gráfica de Confiabilidad para clase de: sin mascarilla

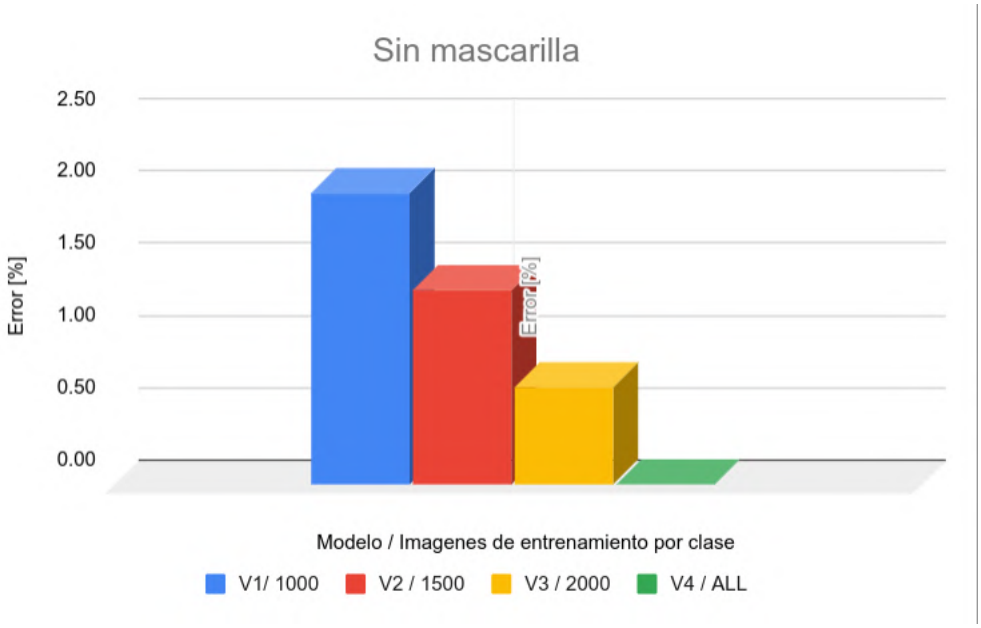

Figura 2.16: Gráfica de Error para clase de: sin mascarilla

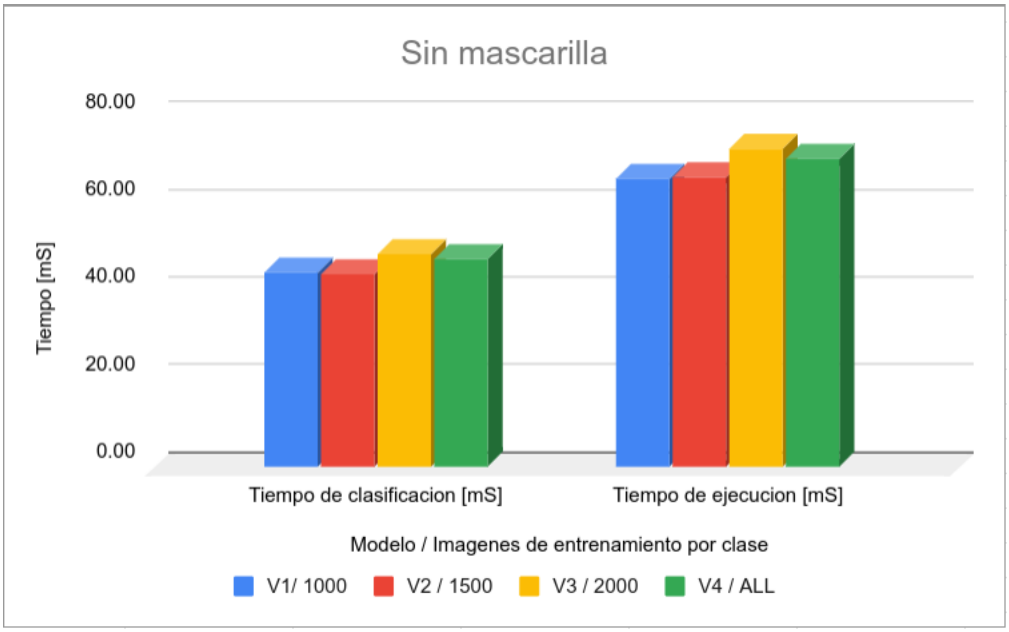

Figura 2.17: Gráfica de tiempos para clase de: sin mascarilla

2.2.1.2. Rendimiento en: Mascarilla mal puesta

Error mascarilla						
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Confiabilidad [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
V1/ 1000	95.33	98.40	4.66	43.62	69.56	11.5
V2 / 1500	100.00	99.72	0.00	43.72	70.08	11.5
V3 / 2000	100.00	99.64	0.00	44.28	70.46	11.5
V4 / ALL	100.00	99.34	0.00	43.98	70.17	11.5

Figura 2.18: Tabla de parámetros de rendimiento para clase de: mascarilla mal puesta

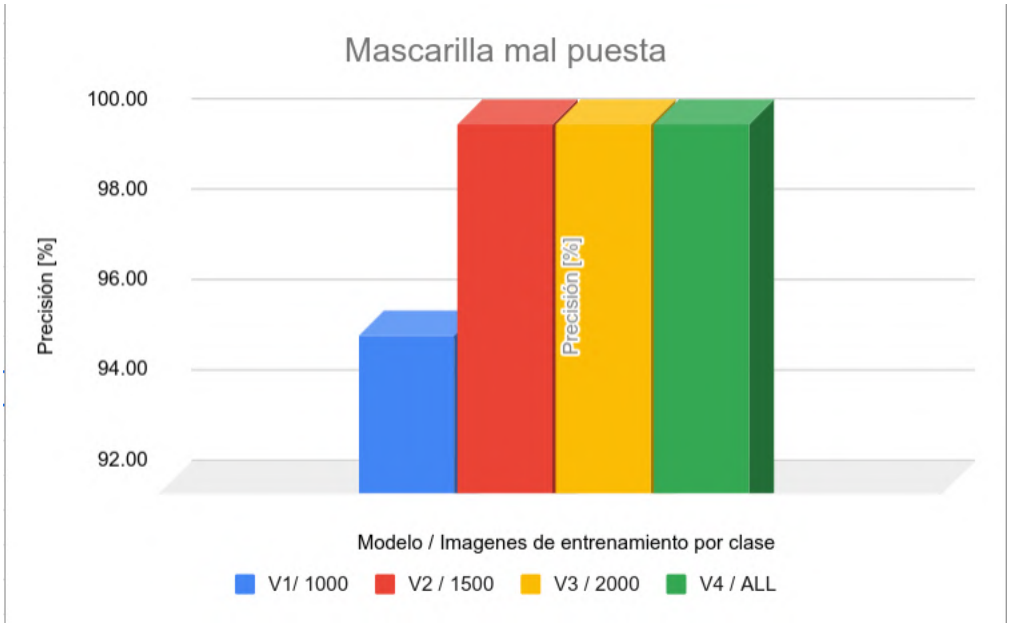

Figura 2.19: Gráfica de precisión para clase de: mascarilla mal puesta

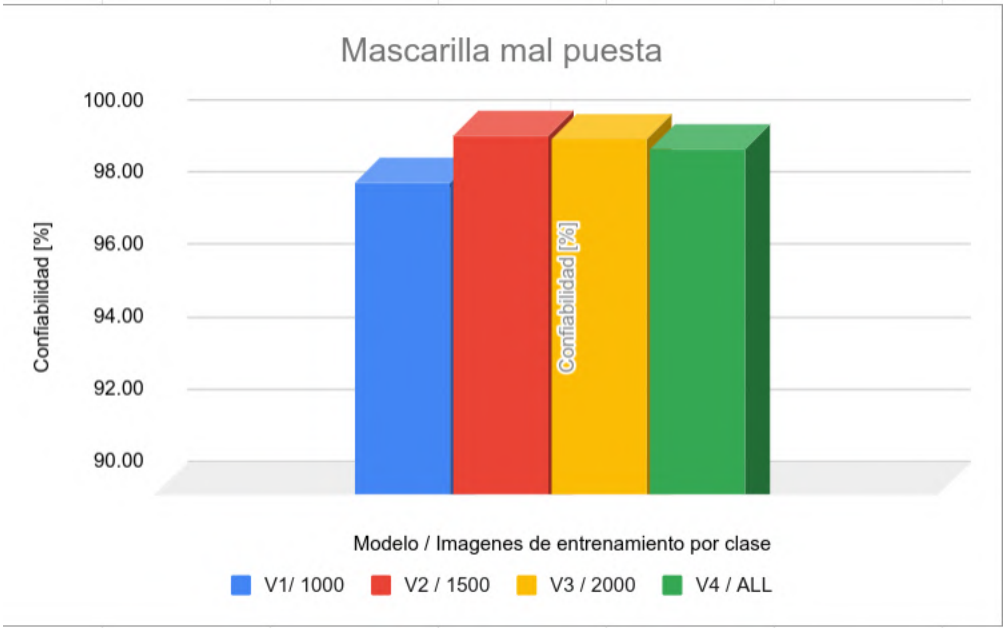

Figura 2.20: Gráfica de confiabilidad para clase de: mascarilla mal puesta

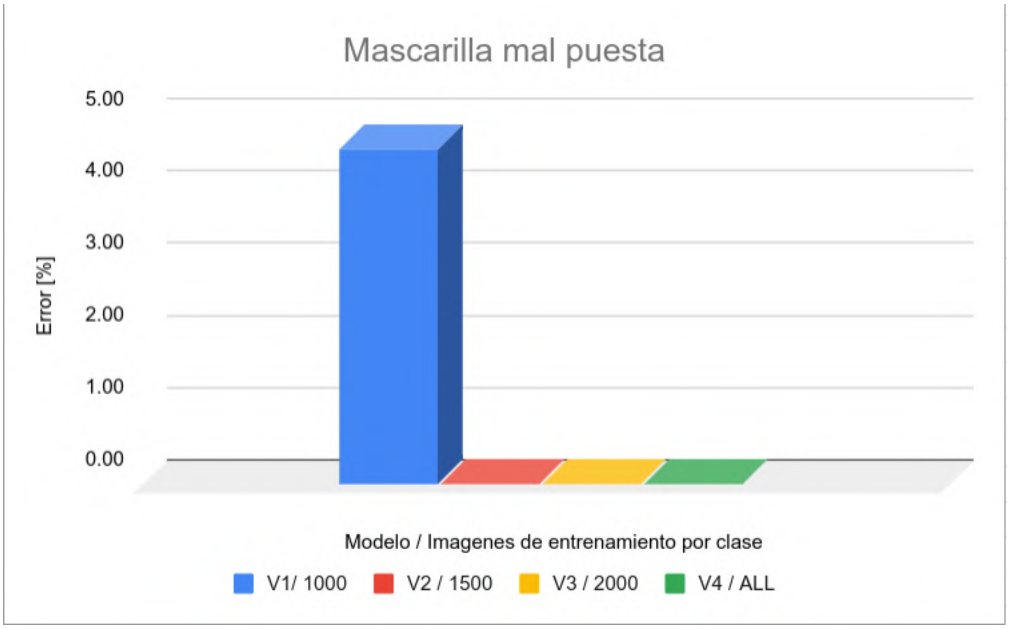

Figura 2.21: Gráfica de Error para clase de: mascarilla mal puesta

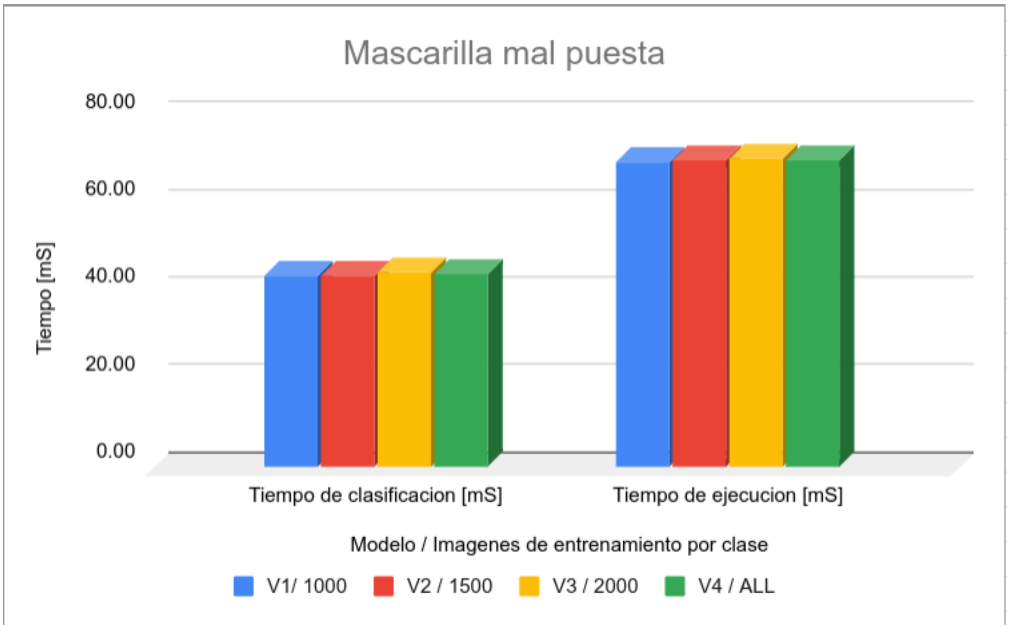

Figura 2.22: Gráfica de tiempos para clase de: mascarilla mal puesta

2.2.1.3. Rendimiento en: con mascarilla

Con mascarilla						
Modelo / Imagenes de entrenamiento por clase	Precisión [%]	Confiabilidad [%]	Error [%]	Tiempo de clasificacion [mS]	Tiempo de ejecucion [mS]	Peso del modelo (MB)
V1/ 1000	56.66	86.30	43.33	43.78	74.89	11.5
V2 / 1500	94.66	99.37	5.33	45.45	76.68	11.5
V3 / 2000	98.00	98.94	2.00	43.56	74.71	11.5
V4 / ALL	99.33	99.55	0.67	46.99	79.91	11.5

Figura 2.23: Tabla de parámetros de rendimiento para clase de: con mascarilla

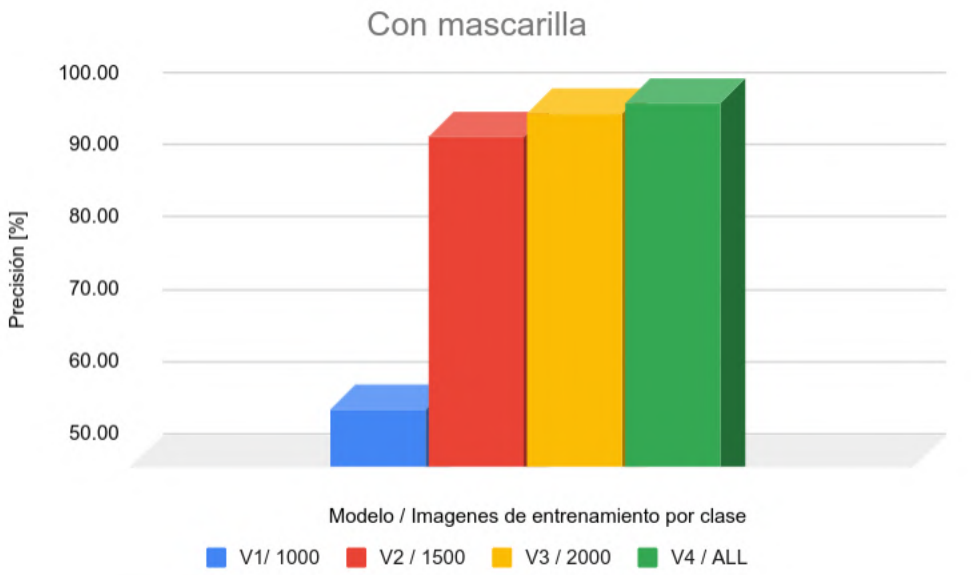

Figura 2.24: Gráfica de precisión para clase de: con mascarilla

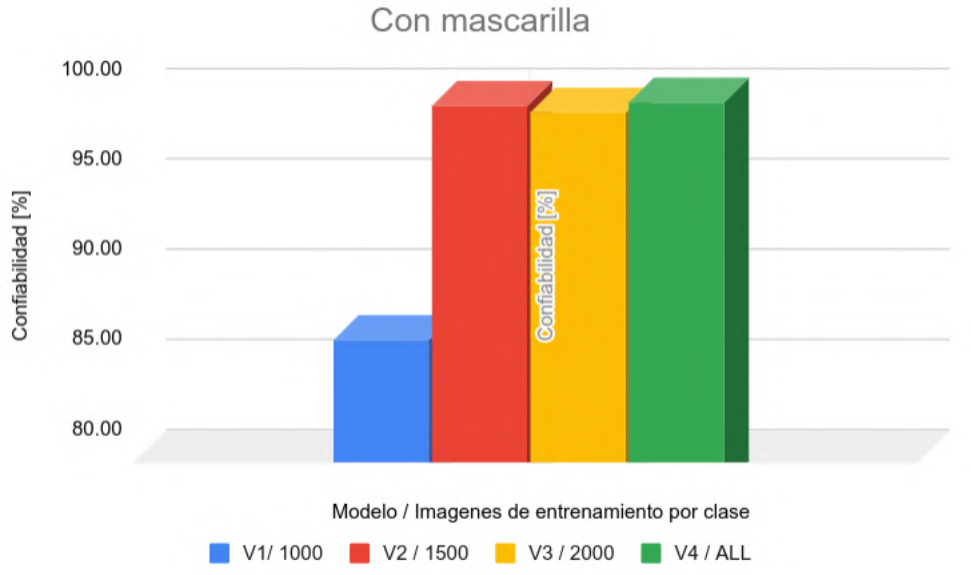

Figura 2.25: Gráfica de confiabilidad para clase de: con mascarilla

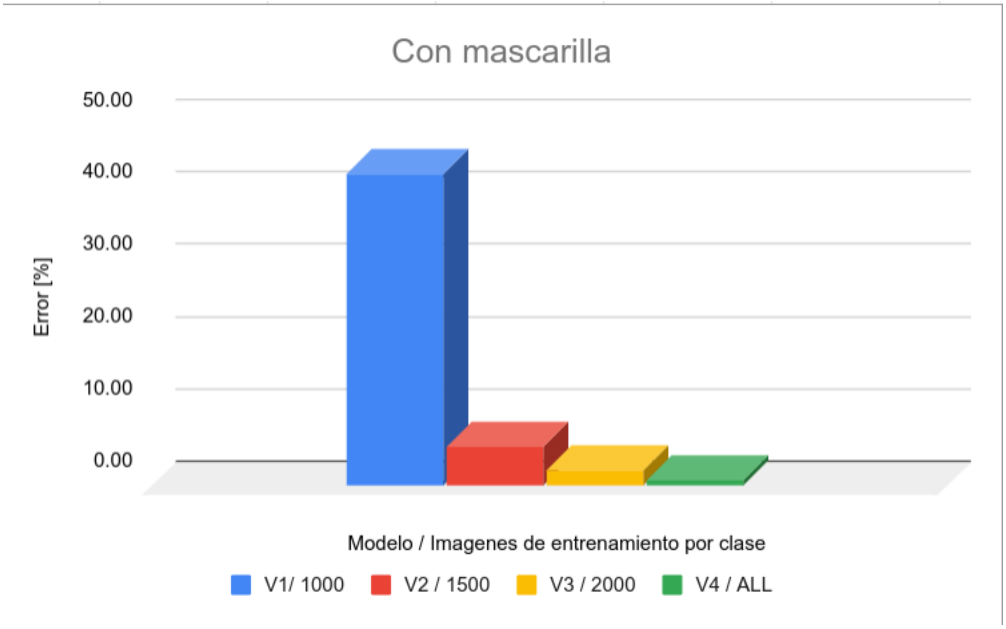

Figura 2.26: Gráfica de Error para clase de: con mascarilla

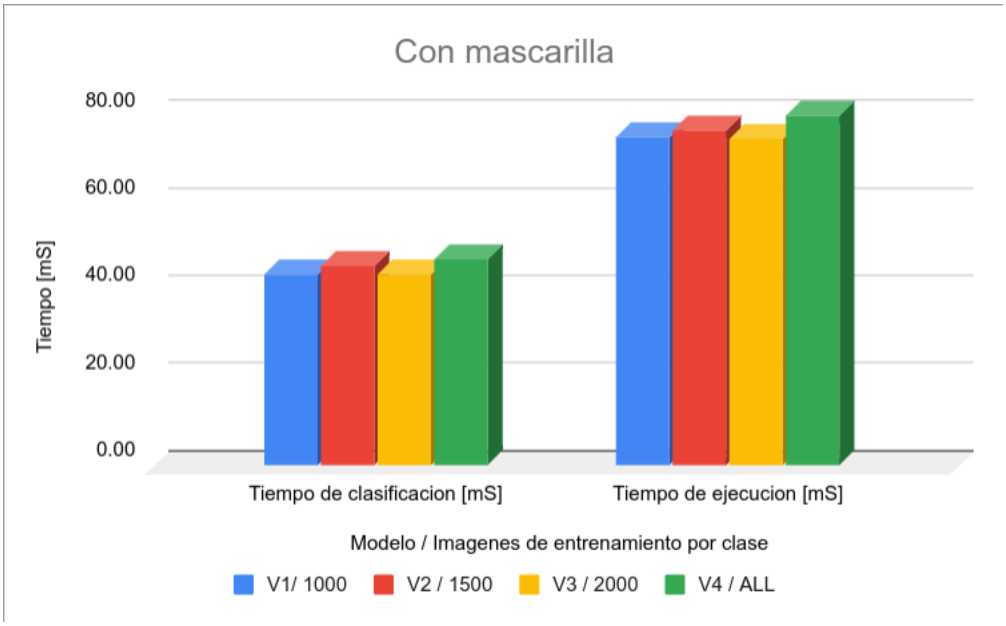

Figura 2.27: Gráfica de tiempos para clase de: con mascarilla

2.2.1.4. Matrices de confusión de la red neuronal en PC.

Matriz de confusión Modelo V1 1000 imágenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	85	65	0
	Mascarilla erronea	7	143	0
	Sin mascarilla	0	3	145

Tabla 2.11: Matriz de confusión para el modelo V1 entrenado con 1000 imágenes por clase.

Matriz de confusión Modelo V2 1500 imágenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	142	8	0
	Mascarilla erronea	0	150	0
	Sin mascarilla	0	2	146

Tabla 2.12: Matriz de confusión para el modelo V2 entrenado con 1500 imágenes por clase.

Matriz de confusión Modelo V3 2000 imágenes		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	147	3	0
	Mascarilla erronea	0	150	0
	Sin mascarilla	0	1	147

Tabla 2.13: Matriz de confusión para el modelo V3 entrenado con 2000 imágenes por clase.

Matriz de confusión Modelo V4 todo el dataset		Predicción		
		Con mascarilla	Mascarilla erronea	Sin mascarilla
Observación	Con mascarilla	149	1	0
	Mascarilla erronea	0	150	0
	Sin mascarilla	0	0	148

Tabla 2.14: Matriz de confusión para el modelo V4 entrenado con todo el dataset.

Al momento de hacer una comparación formal en los resultados obtenidos entre el algoritmo de Machine learning y de Redes neuronales, es que para poder obtener un modelo de redes robusto es necesario entrenar estos modelos con una cantidad mayor de imágenes en comparación con los modelos de LBPH y Eigenface previamente analizados en la sección de Machine learning, el máximo de imágenes con las que se entrenaron los modelos de LBPH y Eigenface fue de 1200 en algunos casos y estos en el caso de LBPH ya representaban una rendimiento considerablemente lento incluso en la plataforma de PC, estos modelos llegaban a alcanzar un peso por encima de los 500 MB, peso que por el contrario en los modelos de la red neuronal en PC no superan los 12 MB, en el caso de los tiempo de ejecución y clasificado en imágenes estos se volvieron a tomar calculando un promedio entre todas imágenes analizadas del dataset de prueba, claramente separados por la clase a analizar; Con mascarilla, Sin mascarilla, Mascarilla mal puesta. En beneficio de la red podemos observar que los valores de precisión para los modelos preparados diferentes a las 1000 imágenes de entrenamiento por clase, obtuvieron precisiones superiores al 90 % en todas las categorías de clasificación, estos modelos funcionan bien en distintas condiciones de luminosidad y resolución en las imágenes tomadas del dataset de prueba así como en los frames tomados de un video streaming o vídeo, esto debido al proceso de data augmentation de la red neuronal convolucional realizado con la base de datos de entrenamiento, que permite cambiar varias propiedades de las imágenes anteriormente mencionadas en el marco teórico al momento de entrenar modelos, objeto que en los mo-

delos de Machine learning proporcionados por la librería de OpenCV para Python no fue posible y se tuvo que optar por entrenar dos clases de modelos unos para imágenes y otros para video streaming en la Raspberry. Si observamos los tiempos de ejecución, podemos ver que en la plataforma de PC es un paso intermedio entre los modelos de Eigenface y LBPH, independientemente de la cantidad de imágenes con la que fue entrenada la red siempre mantiene un estándar de respuesta de ejecución total poco superior a los 70 milisegundos lo que da un promedio de unas 14 a 15 imágenes o fotogramas clasificados por segundo. Considerando su buena precisión y desempeño en distintas condiciones de luminosidad ambiental y resolución, afirmamos una preferencia mayor por estos modelos de redes neuronales que los conseguidos con los clasificadores LBPH y Eigenface.

2.2.2. Errores encontrados en el clasificado de imágenes.

A modo de aclaración estos errores fueron encontrados en el segundo modelo entrenado con la red neuronal convolucional que contiene unas 1500 imágenes por clase, en modelos posteriores la red deja de clasificar estas imágenes de forma errónea.

(a) Error 1

(b) Error 2

Figura 2.28: Errores encontrados dentro de la clase de: con mascarilla

(a) Error 3 (b) Error 4

Figura 2.29: Errores encontrados dentro de la clase de: con mascarilla

Bibliografía

Comunidad AIRX. (s.f.). *Detector de máscaras con OpenCV, Keras / TensorFlow.* https://programmerclick.com/article/42881811018/

Torres & Sánchez. (2022, 17 de enero). *GitHub - davidto098/Facemask-detection-codes: In this repository you will find the codes used in Face mask detection project.* Consultado el 17 de enero de 2022, desde https://github.com/davidto098/Facemask-detection-codes