Análisis y Reporte del desempeño del modelo de ML con Tensorflow

Diego Alfonso Ramírez Montes – A01707596

Data Set.

El data set utilizado tiene por nombre "*DryBeanDataset*" el cual contiene información relativa a la composición y apariencias de varios ejemplos de frijoles así como de su respectiva clase a la cual pertenecen, dicho data set contiene 13,611 instancias.

Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	Class
28395	610.291	208.1781167	173.888747	1.197191424	0.549812187	28715	190.1410973	0.763922518	0.988855999	0.958027126	0.913357755	0.007331506	0.003147289	0.834222388	0.998723889	SEKER
28734	638.018	200.5247957	182.7344194	1.097356461	0.411785251	29172	191.2727505	0.783968133	0.984985603	0.887033637	0.953860842	0.006978659	0.003563624	0.909850506	0.998430331	SEKER
29380	624.11	212.8261299	175.9311426	1.209712656	0.562727317	29690	193.4109041	0.778113248	0.989558774	0.947849473	0.908774239	0.007243912	0.003047733	0.825870617	0.999066137	SEKER
30008	645.884	210.557999	182.5165157	1.153638059	0.498615976	30724	195.4670618	0.782681273	0.976695743	0.903936374	0.928328835	0.007016729	0.003214562	0.861794425	0.994198849	SEKER
30140	620.134	201.8478822	190.2792788	1.06079802	0.333679658	30417	195.896503	0.773098035	0.99089325	0.984877069	0.970515523	0.00669701	0.003664972	0.941900381	0.999166059	SEKER
30279	634.927	212.5605564	181.5101816	1.171066849	0.52040066	30600	196.3477022	0.775688485	0.989509804	0.943851783	0.923725952	0.007020065	0.003152779	0.853269634	0.999235781	SEKER
30477	670.033	211.0501553	184.0390501	1.146768336	0.489477894	30970	196.9886332	0.762401501	0.984081369	0.853079869	0.933373552	0.006924899	0.003242016	0.871186188	0.999048736	SEKER
30519	629.727	212.9967551	182.7372038	1.165590535	0.513759558	30847	197.1243203	0.770681818	0.989366875	0.967109244	0.925480392	0.006979152	0.003158285	0.856513956	0.99834456	SEKER
30685	635.681	213.5341452	183.1571463	1.165852108	0.51408086	31044	197.659696	0.771561479	0.988435769	0.954239808	0.925658498	0.00695891	0.00315155	0.856843654	0.998952981	SEKER
30834	631.934	217.2278128	180.8974686	1.2008339	0.553642225	31120	198.1390121	0.783682806	0.990809769	0.97027823	0.91212543	0.007045074	0.00300804	0.8319728	0.999061142	SEKER
30917	640.765	213.5600894	184.4398709	1.157884618	0.504102365	31280	198.4055115	0.770805285	0.988395141	0.94625818	0.929038343	0.006907529	0.00317422	0.863112242	0.999384389	SEKER
31091	638.558	210.4862549	188.3268476	1.117664622	0.446621924	31458	198.9630385	0.786377317	0.988333651	0.958172836	0.945254305	0.006770006	0.003333984	0.8935057	0.998639711	SEKER
31107	640.594	214.6485485	184.9692526	1.160455295	0.507365875	31423	199.0142269	0.761046142	0.989943672	0.952581757	0.927163162	0.006900329	0.003145388	0.859631529	0.997563959	SEKER
31158	642.626	216.4848362	183.6443122	1.178826797	0.529514251	31492	199.1773023	0.798759229	0.989394132	0.948119004	0.92005198	0.00694797	0.003071052	0.846495646	0.997871751	SEKER
31158	641.105	212.0669751	187.1929601	1.132879009	0.469924157	31474	199.1773023	0.781313473	0.989959967	0.952623101	0.939218858	0.006806181	0.00326701	0.882132064	0.999348898	SEKER
31178	636.888	212.9759252	186.5620882	1.141582018	0.482352224	31520	199.2412169	0.764110482	0.989149746	0.965899596	0.935510513	0.006830968	0.003227429	0.875179919	0.999089658	SEKER
31202	644.454	215.6406947	184.4716842	1.168963657	0.517871223	31573	199.3178875	0.779192888	0.988249454	0.944079243	0.924305534	0.006911118	0.003111647	0.85434072	0.998693253	SEKER
31203	639.782	215.067737	184.8748759	1.163315112	0.510946829	31558	199.3210815	0.762984155	0.988750871	0.957948542	0.926782809	0.006892534	0.003136682	0.858926376	0.999202033	SEKER
31272	638.666	212.4503189	187.535939	1.132851229	0.469883494	31593	199.5413417	0.770322199	0.989839521	0.963425036	0.939237666	0.006793627	0.003261246	0.882167393	0.999364415	SEKER
31335	635.011	216.7900923	184.1634403	1.177161395	0.52758671	31599	199.7422367	0.774277242	0.991645305	0.976510834	0.921362386	0.006918465	0.003075469	0.848908647	0.999302487	SEKER
31374	636.401	219.865394	182.0088637	1.207992784	0.56099452	31604	199.8664991	0.769196823	0.99272244	0.973459862	0.909040279	0.007007885	0.002951884	0.826354229	0.998229946	SEKER
31530	638.857	213.7856543	188.0664823	1.136755746	0.475535642	31791	200.3627781	0.768949371	0.991790129	0.970792739	0.937213391	0.006780389	0.003226921	0.878368941	0.99849094	SEKER
31573	674.103	217.3070261	185.4482507	1.171793345	0.52126839	32197	200.499357	0.756964757	0.980619312	0.873118507	0.922654737	0.006882685	0.003076767	0.851291764	0.997538024	SEKER
31637	656.711	229.7192546	175.510446	1.308863717	0.645190802	32045	200.702465	0.761823348	0.987267905	0.921842182	0.873685862	0.007261095	0.002609775	0.763326986	0.999090996	SEKER
31675	657.431	236.7526321	171.2105592	1.3828156	0.690678219	32009	200.8229633	0.740935673	0.989565435	0.92092896	0.848239622	0.007474432	0.002386889	0.719510457	0.994950284	SEKER
31682	646.721	210.0456816	192.2484159	1.092574316	0.402841974	32026	200.8451524	0.773184303	0.989258727	0.951893863	0.956197485	0.006629811	0.003418781	0.91431363	0.998955589	SEKER
31703	656.305	215.7089067	187.2724497	1.151845384	0.496263281	32093	200.9117052	0.777110501	0.987847817	0.92490856	0.931401991	0.006804053	0.003158611	0.867509669	0.999237008	SEKER
31748	641.826	219.7765183	184.1151053	1.193690859	0.546072628	32020	201.0542441	0.775589974	0.991505309	0.968482156	0.914812218	0.006922531	0.002990698	0.836881394	0.99897767	SEKER

Figura 1. Ejemplo del Dataset.

Modelo de Machine Learning

Para el modelo de Machine Learning se decidió por una red neuronal entrenada con ayuda de Tensorflow. Para el mismo fue necesario realizar "*One-Hot Encoding*" a fin de poder predecir las clases de forma adecuada, a si mismo se realizó el Split de los datos en 80% training y 20% para el testeo. Así mismo fue necesario estandarizar los datos a fin de que el modelo sea capaz de converger, pues el mismo posee algunos datos que son demasiado grandes y difieren de los más pequeños, lo cual dificulta, o impide, la convergencia del modelo.

Se decidió por utilizar 2 capas ocultas de neuronas de 64 y 32 neuronas cada uno con la función sigmoide como función de activación, mientras que la capa final

posee las 7 clases con la función de activación softmax. Se decidió utilizar cross-entropy en su versión categórica como función de pérdida, pues la misma es la más adecuada en una situación de categorización.

El número de batches fue elegido realmente de forma caprichosa en 100.

```python
# Define the neural network model using TensorFlow
model = models.Sequential([
    layers.Dense(64, activation='sigmoid', input_shape=(X_train_normalized.shape[1],)),
    layers.Dense(32, activation='sigmoid'),
    layers.Dense(7, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
epochs = 100
history = model.fit(X_train_normalized, y_train, epochs=epochs, validation_data=(X_test_normalized, y_test))

# Evaluate the model
train_loss, train_accuracy = model.evaluate(X_train_normalized, y_train)
test_loss, test_accuracy = model.evaluate(X_test_normalized, y_test)

# Confusion matrix
y_pred = model.predict(X_test_normalized)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)
confusion_mtx = confusion_matrix(y_true_classes, y_pred_classes)

# Extract final loss, accuracy and val_accuracy values
final_loss = history.history['loss'][-1]
final_accuracy = history.history['accuracy'][-1]
final_val_accuracy = history.history['val_accuracy'][-1]
```

Figura 2. Modelo utilizado.

Evaluación del Modelo.

A fin de poder evaluar de forma correcta el modelo existen varias herramientas, en este caso estaremos haciendo uso de las siguientes:

- **Matriz de confusión**: Una matriz de confusión nos permite evaluar cuantas veces nuestro modelo escogió una cierta opción contra la cantidad de veces en que dicha opción fue la correcta, en nuestro caso la misma luciría así:

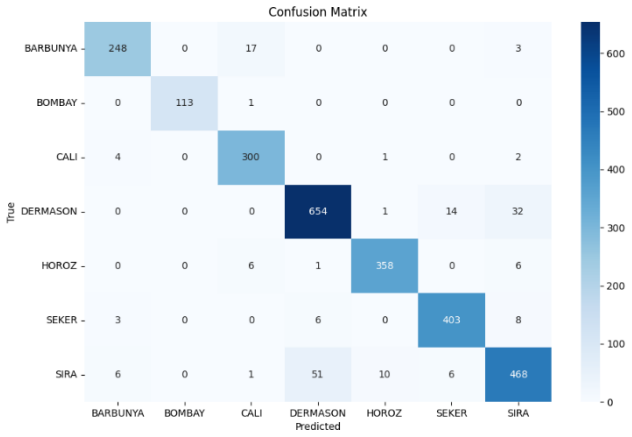

Figura 3. Matriz de confusión.

- **Accuracy**: El parámetro de accuracy en nuestra matriz de confusión nos permite saber la cantidad de veces que nuestro modelo acertó. Dicho valor varía entre [1,0], siendo que en nuestro caso es de 0.93426.
- **Precisión**: En este caso sería la cantidad de veces que nuestro modelo predijo la clase y resulto ser la correcta, nuevamente dicho parámetro oscila entre [0,1], siendo que el nuestro es de 0.93418.
- **Recall**: Mide cuántas veces el modelo identificó correctamente todos los casos relevantes de una clase. Para nuestro caso sería 0.93426.
- **F1**: Es la media armónica de 'precision' y 'recall'. Nos ayuda a evaluar la eficacia general de nuestro modelo de clasificación. Para nosotros tendría un valor de 0.93402.

Estos valores nos permiten medir de forma objetiva que tan bien se desempeñó nuestro modelo en general, sin embargo, también podemos extender y complementar nuestro análisis general con ayuda de otras gráficas e ideas.

Primeramente sería importante denotar que nuestro modelo, tal cual esta, no posee un paso de validación, el cual puede ser importante de añadir para tener una forma extra de medir la precisión/error/varianza de nuestro modelo. Una vez que añadimos el mismo obtenemos las siguientes gráficas:

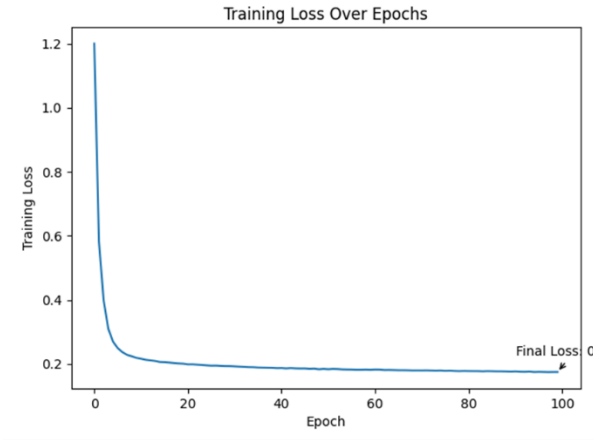

Figura 4. Gráfica de la pérdida a través de los epochs.

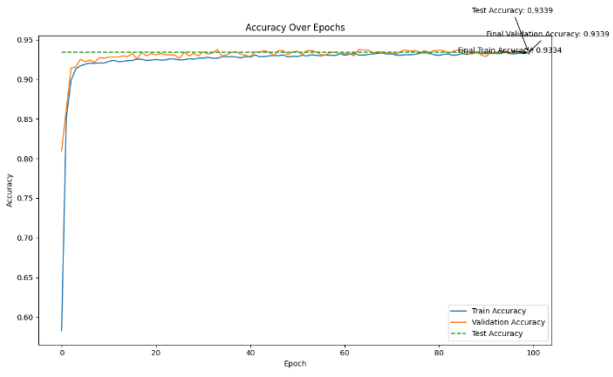

Figura 5. Gráfica del accuracy de train, test y validation.

En general podemos observar que, en caso de tener over o underfitting, las gráficas deberían de tener valores demasiado buenos en el caso de train y tener un

desempeño cercano a pésimo o mediocre en el caso de la validación y el testeo, sin embargo, podemos observar que este no es el caso al tener gráficas y valores similares en cada caso esto, una vez más, nos indica de forma objetiva que nuestro modelo no tiene un fitt realista, siendo que posee una baja varianza así como un bias bajo.

Mejoras.

De forma realista no es necesario realizar mejoras en este modelo tal cual, pues como se puede observar del análisis anterior, el mismo posee un fitt adecuado el cual podría arruinarse si tratamos de volver más complejo el modelo, e.g. Añadiendo más neuronas o capas ocultas, por tanto no es del todo necesario intentar implementar mejoras. A fin de documentar en el análisis, he de decir que se realizó una “mejora” al modelo al cambiar de función de activación durante el desarrollo de este, sin embargo, esta sólo fue de 0.01 por lo que, como comento, dichas mejoras no son necesarias ni contribuirían de forma apreciable al desempeño de este, el cual, ya es adecuado.

Conclusiones.

Tal y como se pudo apreciar por medio de funciones y pruebas de carácter objetivo, el modelo presentado en este trabajo realmente no necesita mejoras sustanciales pues, a través de las demostraciones presentadas, se puede observar que el modelo posee un fitt adecuado y una precisión apreciable cercana al 94% la cual podría ser arruinada al realizar aumentos de complejidad al modelo o alguna mejora drástica del mismo. Realmente es que no se descartan mejoras pues, así como se mencionó, es posible realizar cambios menores que provean de una mejora menor, solo eso. Con este trabajo se puede apreciar de forma más clara la necesidad de contar con métricas que nos ayuden a señalar los defectos y/o fortalezas que nuestro modelo posee, así como tener pruebas objetivas de las áreas de oportunidad y la validez del mismo.