

Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ciência da Computação
Inteligência Artificial Conexionista

Augusto André Souza Berndt 202000735

Exercício 6 - Cat vs Non-Cat

Enunciado: Resolver o problema com um perceptron apenas (regressão logística), uma rede de camada rasa e uma rede convolucional. Um Relatório completo deve ser entregue, contendo informações de **Quantos e Quais experimentos foram feitos até chegar no resultado final**; Como foi o treinamento; Qual a taxa de acertos da rede; A matriz de confusão, etc...

Introdução: A cada página deste documento se realiza um experimento diferente. Cada experimento é numerado. Se realizou um total de 9 experimentos diferentes, tentando sempre reduzir o número de exemplares errados no conjunto de teste. O último experimento foi o que obteve o menor número de exemplares errados no teste, com um total de somente 3 exemplares errados.

1. Resultados Originais - Primeiramente armazenamos aqui as saídas originais como já disponibilizadas no código para futuras comparações. A Figura 1 demonstra que talvez a rede esteja sendo treinada com muitas épocas, uma vez que o loss da validação começa a se afastar do treino, próximo da época 125. Podemos ver também que a rede demonstrou uma ótima capacidade de generalização, confundindo apenas 9 exemplares.

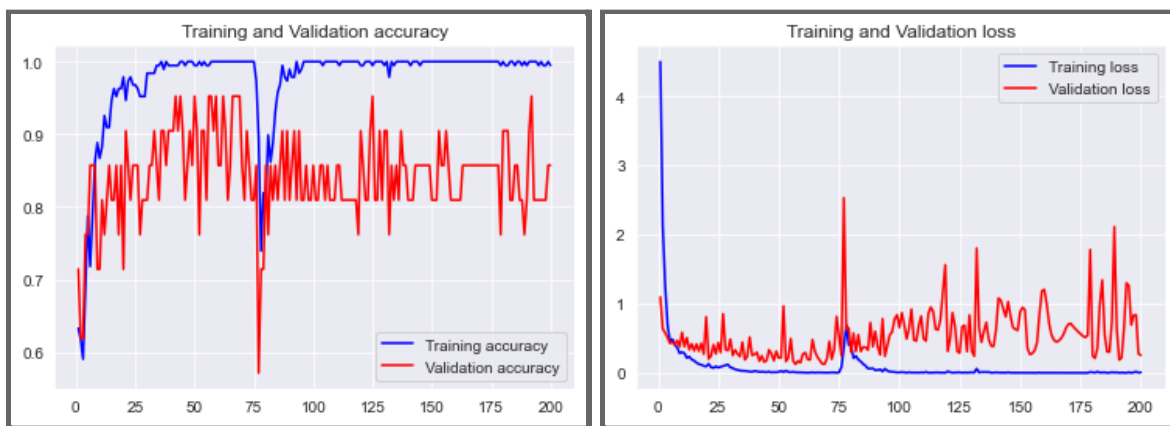


Figura 1 (treinamento com valores originais).

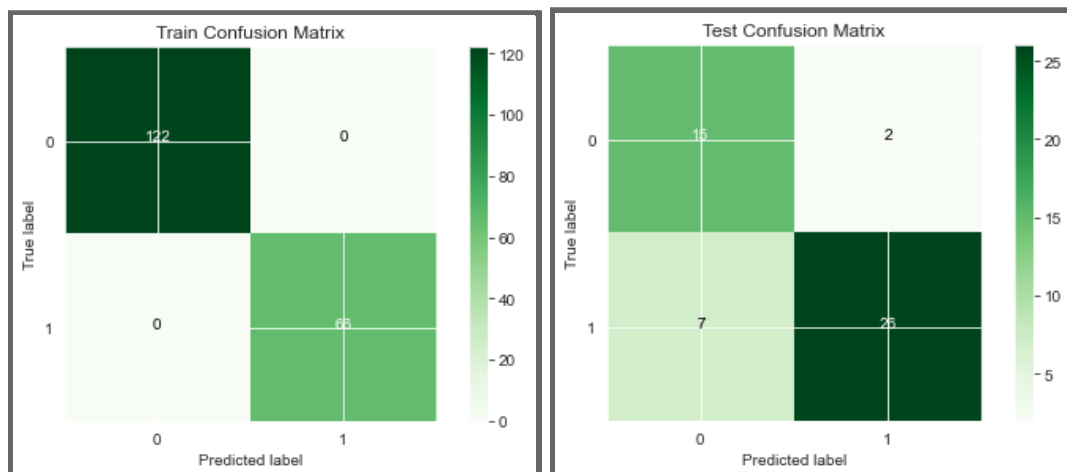


Figura 2 (matriz de confusão com valores originais).

2. “Resolver o problema com um perceptron apenas (regressão logística)”: a regressão logística é feita com a função de ativação do tipo *sigmoid* em uma camada completamente conectada (*dense*) com apenas um neurônio, a camada de saída é mantida sem função de ativação para manter os formatos já esperados no resto do código. Não consegui deixar de usar a função *flatten*, sem ela os shapes sempre eram mantidos como (64,64,n) sendo n o número de neurônios da camada, quando o esperado seria como na Figura 4. O treinamento não se mostrou nada promissor, não tenho certeza se algo está errado, mas com um único perceptron o modelo não foi capaz de classificar as imagens. O formato printado pelo “*model.summary*” parece estar como esperando, uma única camada interna com apenas 1 neurônio e 12288 (=64*64*3) entradas e uma camada de saída com 2 neurônios sem ativação.

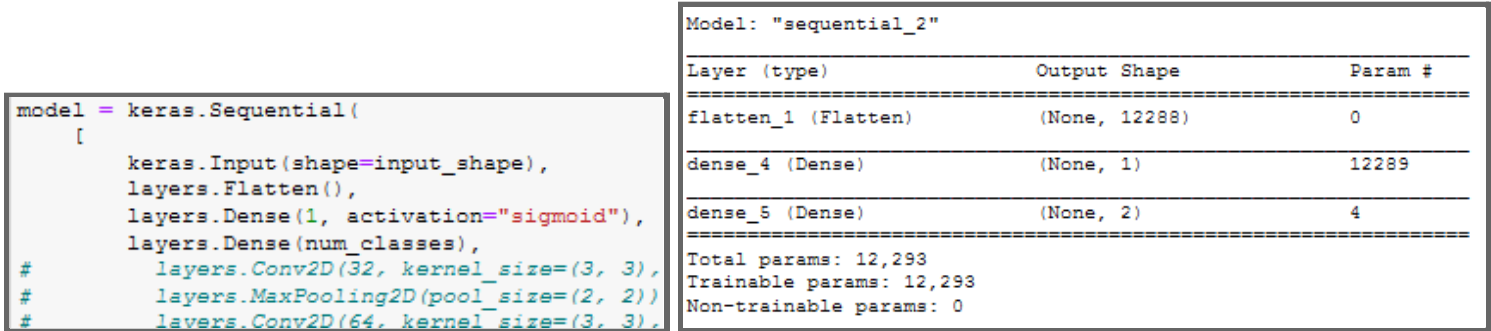


Figura 3 (modelo com único perceptron).

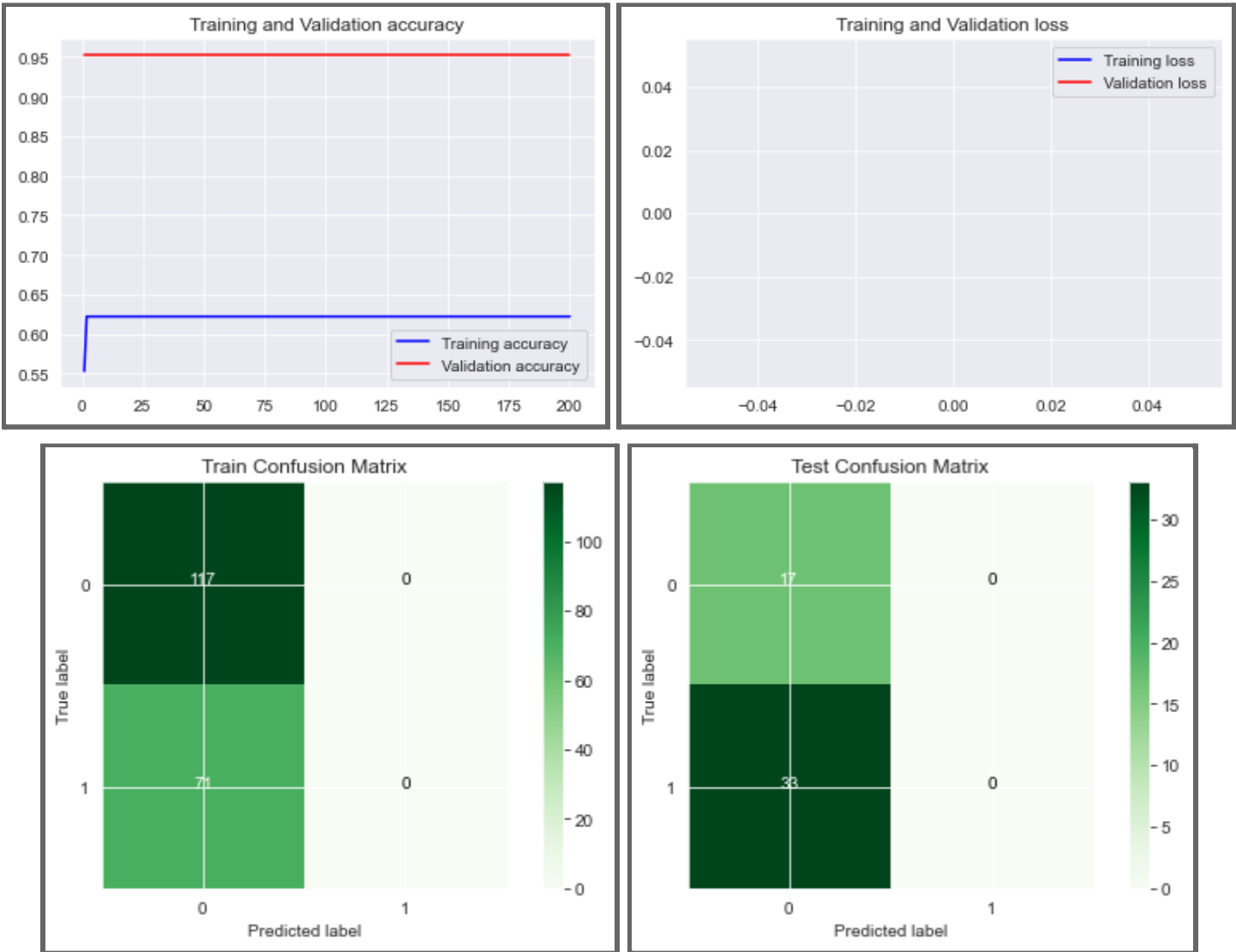


Figura 4 (treinamento de rede com 1 perceptron).

3. “Resolver o problema com uma rede de camada rasa”: Para isto utilizamos apenas uma camada interna com 50 neurônios e ativação ReLu, também se utiliza uma camada de saída com apenas dois neurônios e ativação *softmax*, como consta na Figura 5. Com 200 épocas de treinamento, o modelo já apresentou um treinamento bem bom, mas com um pequeno overfitting, uma vez que alguns indivíduos foram confundidos no teste e acertou todos no treinamento, como mostram as matrizes de confusão na Figura 6. A acurácia ao longo do treinamento demonstra o overfitting de maneira mais explícita.

```
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Dense(50, activation="relu"),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="softmax"),
# layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(256, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(512, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1024, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2048, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4096, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(8192, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(16384, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(32768, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(65536, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(131072, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(262144, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(524288, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1048576, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2097152, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4194304, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(8388608, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(16777216, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(33554432, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(67108864, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(134217728, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(268435456, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(536870912, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1073741824, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2147483648, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4294967296, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(8589934592, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(17179869184, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(34359738368, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(68719476736, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(137438953472, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(274877906944, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(549755813888, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1099511627776, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2199023255552, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4398046511104, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(8796093022208, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(17592186044416, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(35184372088832, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(70368744177664, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(140737488355328, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(281474976710656, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(562949953421312, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1125899906842624, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2251799813685248, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4503599627370496, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(9007199254740992, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(18014398509481984, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(36028797018963968, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(72057594037927936, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(144115188075855872, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(288230376151711744, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(576460752303423488, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1152921504606846976, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2305843009213693952, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4611686018427387904, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(9223372036854775808, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(18446744073709551616, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(36893488147419103232, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(73786976294838206464, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(147573952589676412928, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(295147905179352825856, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(590295810358705651712, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1180591620717411303424, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2361183241434822606848, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4722366482869645213696, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(9444732965739290427392, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(18889465931478580854784, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(37778931862957161709568, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(75557863725914323419136, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(151115727451828646838272, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(302231454903657293676544, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(604462909807314587353088, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1208925819614629174706176, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2417851639229258349412352, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4835703278458516698824704, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(9671406556917033397649408, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(19342813113834066795298816, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(38685626227668133590597632, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(77371252455336267181195264, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(154742504910672534362390528, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(309485009821345068724781056, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(618970019642690137449562112, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1237940039285380274899124224, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2475880078570760549798248448, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(4951760157141521099596496896, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(9903520314283042199192993792, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(19807040628566084398385987584, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(39614081257132168796771975168, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(79228162514264337593543950336, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(158456325028528675187087900672, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(316912650057057350374175801344, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(633825300114114700748351602688, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1267650600228229401496703205376, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2535301200456458802993406410752, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5070602400912917605986812821504, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(10141204801825835211973625643008, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(20282409603651670423947251286016, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(40564819207303340847894502572032, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(81129638414606681695789005144064, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(162259276829213363391578010288128, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(324518553658426726783156020576256, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(649037107316853453566312041152512, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1298074214633706907132624082305024, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2596148429267413814265248164610048, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5192296858534827628530496329220096, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(10384593717069655257060992658440192, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(20769187434139310514121985316880384, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(41538374868278621028243970633760768, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(83076749736557242056487941267521536, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(166153499473114484112975882535043072, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(332306998946228968225951765070086144, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(664613997892457936451903530140172288, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1329227995784915872903807060280344576, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2658455991569831745807614120560689152, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5316911983139663491615228241121378304, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(10633823966279326983230456482242756608, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(21267647932558653966460912964485513216, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(42535295865117307932921825928971026432, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(85070591730234615865843651857942052864, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(170141183460469231731687303715884105728, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(340282366920938463463374607431768211456, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(680564733841876926926749214863536422912, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1361129467683753853853498429727072845824, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2722258935367507707706996859454145691648, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5444517870735015415413993718908291383296, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(10889035741470030830827987437816582766592, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(21778071482940061661655974875633165533184, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(43556142965880123323311949751266331066368, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(87112285931760246646623899502532662132736, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(174224571863520493293247799005065324265472, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(348449143727040986586495598010130648530944, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(696898287454081973172991196020261297061888, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1393796574908163946345982392040522594123776, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2787593149816327892691964784081045188247552, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5575186299632655785383929568162090376495104, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(11150372599265311570767859136324180752990208, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(22300745198530623141535718272648361505980416, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(44601490397061246283071436545296723011960832, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(89202980794122492566142873090593446023921664, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(178405961588244985132285746181186892047843328, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(356811923176489970264571492362373784095686656, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(713623846352979940529142984724747568191373312, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1427247692705959881058285969449495136382746624, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2854495385411919762116571938898990272765493248, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5708990770823839524233143877797980545530986496, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(11417981541647679048466287755595961091061972992, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(22835963083295358096932575511191922182123945984, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(45671926166590716193865151022383844364247891968, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(91343852333181432387730302044767688728495783936, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(182687704666362864775460604089535377456991567872, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(365375409332725729550921208179070754913983135744, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(730750818665451459101842416358141509827966271488, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1461501637330902918203684832716283019655932542976, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2923003274661805836407369665432566039311865085952, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5846006549323611672814739330865132078623730171904, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(11692013098647223345629478661730264157247460343808, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(23384026197294446691258957323460528314494920687616, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(46768052394588893382517914646921056628989841375232, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(93536104789177786765035829293842113257979682750464, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(187072209578355573530071658587684226515959365500928, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(374144419156711147060143317175368453031918731001856, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(748288838313422294120286634350736906063837462003712, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1496577676626844588240573268701473812127674924007424, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(2993155353253689176481146537402947624255349848014848, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(5986310706507378352962293074805895248510699696029696, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(11972621413014756705924586149611790497021399392059392, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(23945242826029513411849172299223580994042798784118784, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(47890485652059026823698344598447161988085597568237568, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(95780971304118053647396689196894323976171195136475136, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(191561942608236107294793378393788647952342390272950272, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(383123885216472214589586756787577295904684780545900544, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(766247770432944429179173513575154591809369561091801088, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1532495540865888858358347027150309183618739122183602176, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(3064991081731777716716694054300618367237478244367204352, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(6129982163463555433433388108601236734474956488734408704, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(12259964326927110866866776217202473468949912977468817408, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(24519928653854221733733552434404946937899825954937634816, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(49039857307708443467467104868809893875799651909875269632, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(98079714615416886934934209737619787751599303819750539264, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(196159429230833773869868419475239575503198607639501078528, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(392318858461667547739736838950479151006397215279002157056, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(784637716923335095479473677900958302012794430558004314112, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(1569275433846670190958947355801916604025588861116008628224, kernel_size=(3, 3), activation="relu"),
# layers.MaxPooling2D(pool_size=(2, 2)),
# layers.Conv2D(3138550867693340381917894711603833208051177722232017256448, kernel_size=(3,
```

4. “Resolver o problema com um perceptron apenas (regressão logística), uma rede de camada rasa e uma rede convolucional” (150 épocas): Para este desafio tentei ajustar a rede já disponibilizada pelo professor com diversas camadas convolucionais. Um treinamento somente fazendo menos épocas já demonstra resultados melhores que os originais. O modelo original erra 9 exemplares, já o modelo atual erra 6 exemplares. Esta melhora mostra que talvez seja possível encontrar uma configuração de treinamento que acerte todos, ou quase todos, exemplares de teste. A Figura 8 mostra uma boa evolução no treinamento, a função custo avança junto entre validação e treino, o que é um bom indicador. No entanto a acurácia demonstra um certo afastamento entre treino e validação, talvez seja interessante um treinamento com ainda menos épocas.

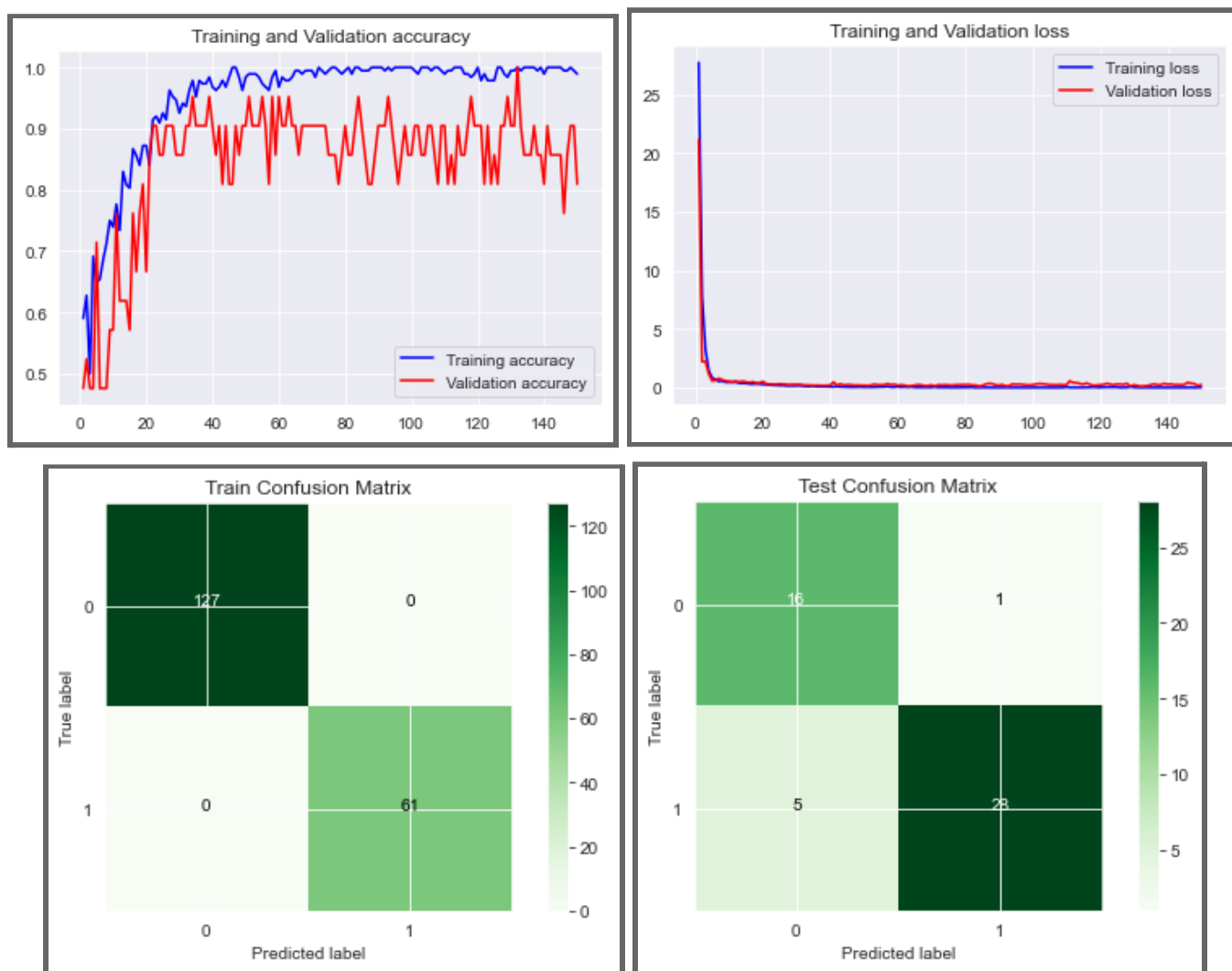


Figura 8 (treinamento com rede convolucional - 150 épocas).

5. 110 épocas: A partir daqui irei tentar ajustar a rede já disponibilizada na intenção de melhorar os resultados de treinamento. O objetivo é reduzir o número de exemplares errados no conjunto de teste. Primeiro tentamos somente reduzir um pouco mais o número de épocas. A arquitetura se mantém a mesma, assim como no experimento anterior, só alterei o números de épocas do treinamento. Desta vez temos 10 exemplares errados no teste, ou seja, pior do que o original.

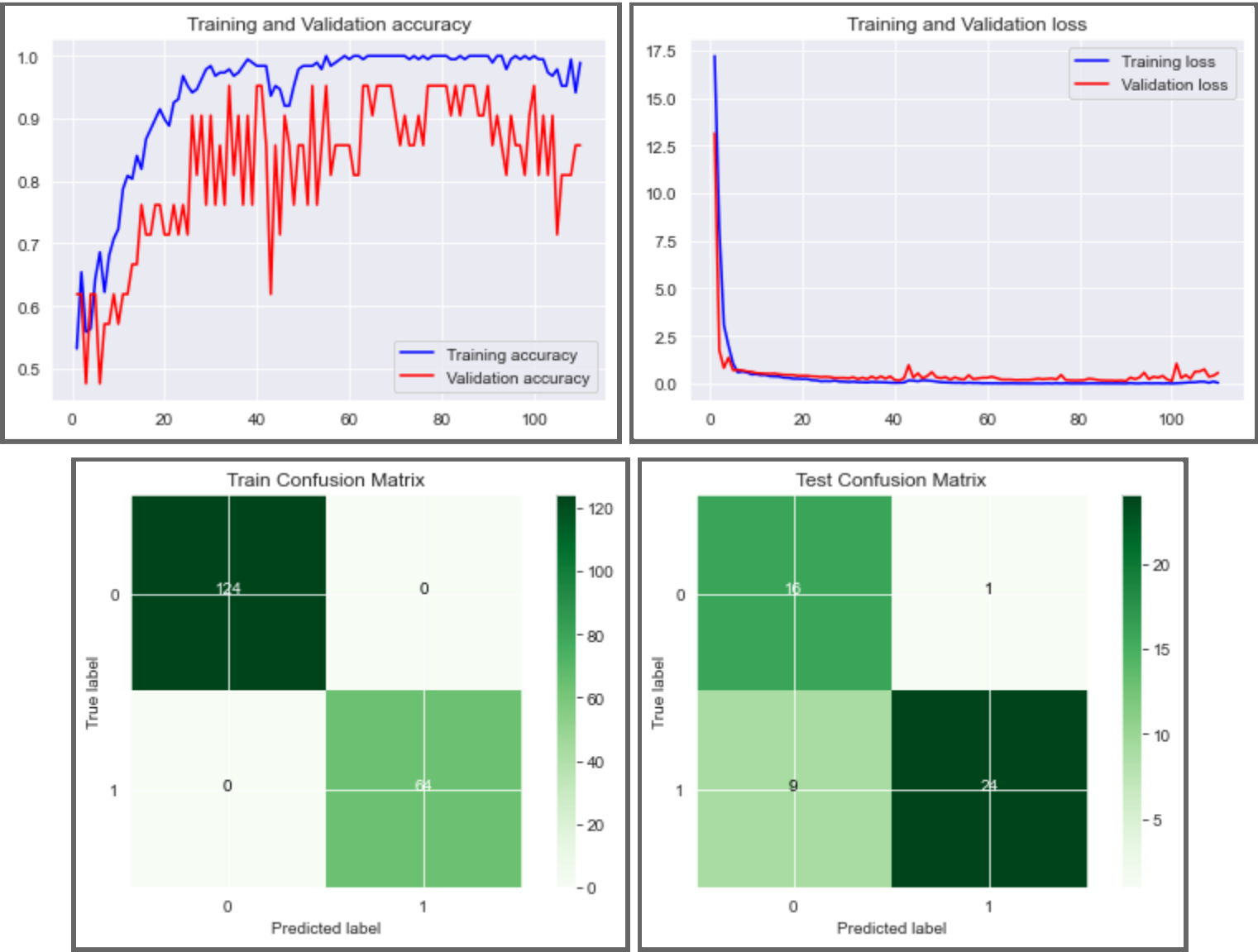


Figura 9 (treinamento com **110 épocas**).

6. 125 épocas: Desta vez temos 7 exemplares errados no conjunto de teste. Podemos ver que o número ótimo de épocas é próximo de 150. A partir de agora tentarei modificar a arquitetura da rede, mantendo 150 épocas, uma vez que até o momento foi o melhor experimento.

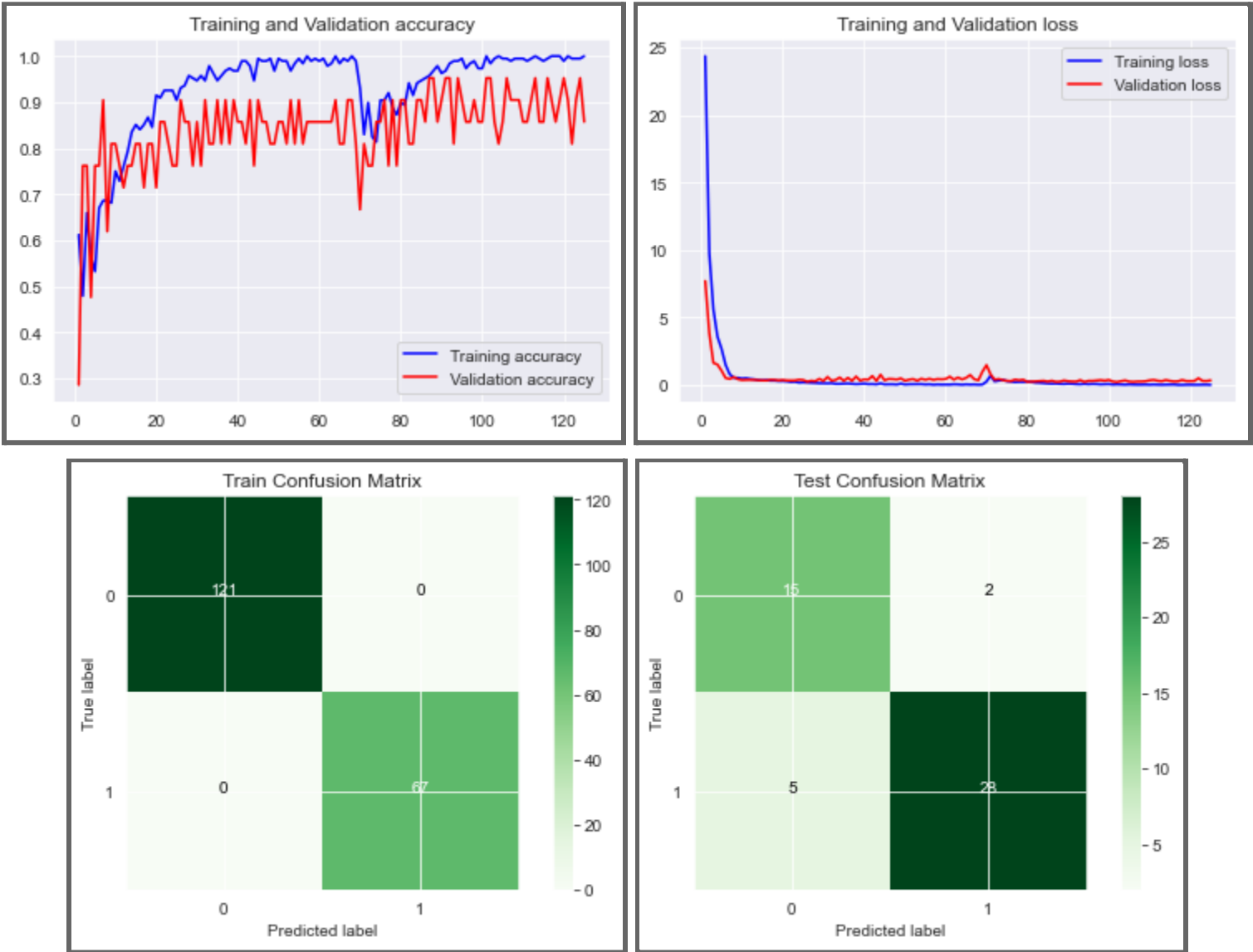


Figura 10 (treinamento com **125 épocas**).

7. Substituindo alguns ReLu's por SeLu (exercício anterior [5] foram melhores do que ReLu) com 150 épocas: Obtivemos 8 exemplares errados intercalando entre ReLu e SeLu. Ainda não foi melhor do que o experimento 4 que obteve somente 6 exemplares errados no teste.

```
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="softmax"),
])
```

Figura 11 (modelo ReLu e SeLu).

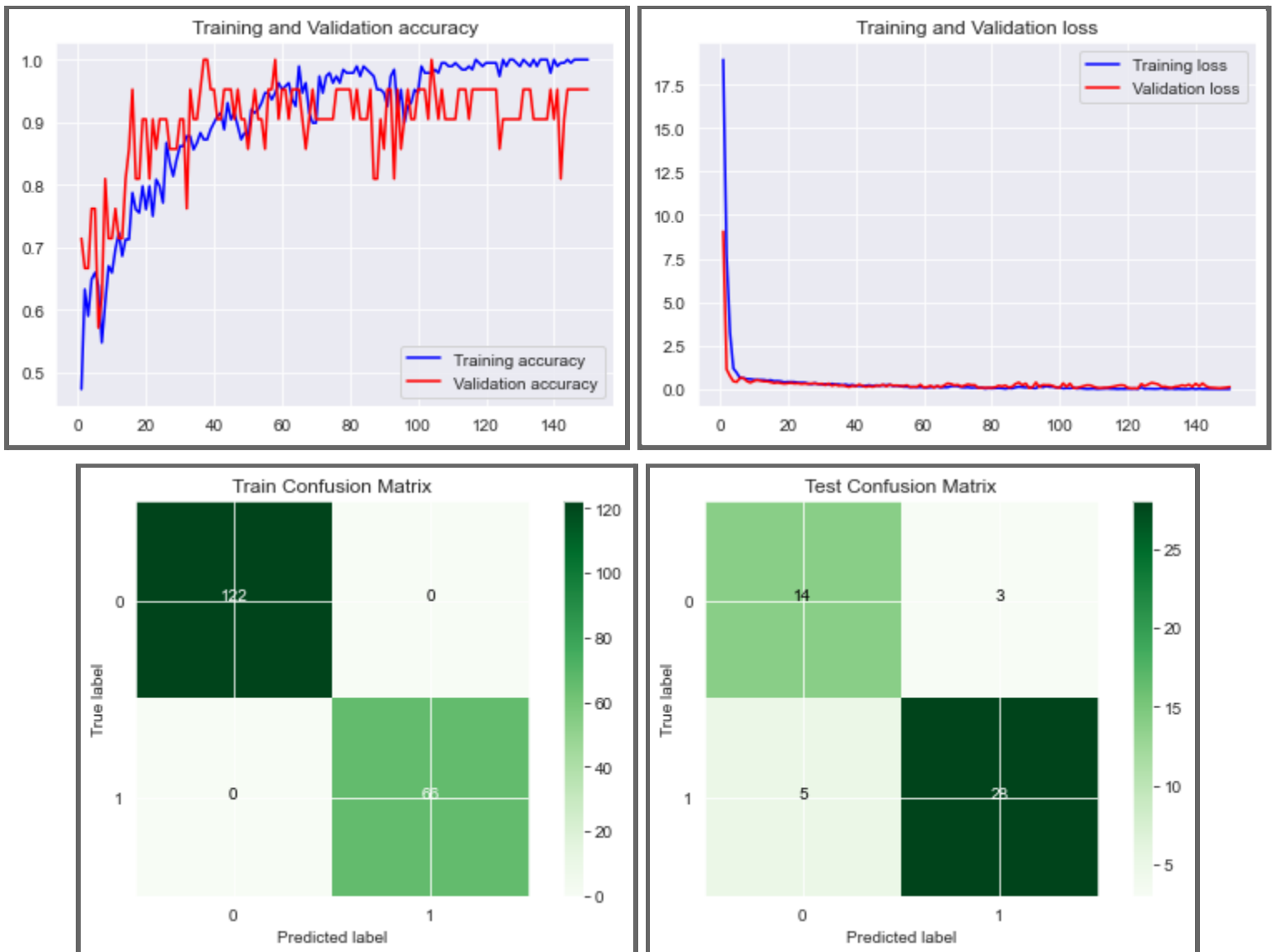


Figura 12 (treinamento com modelo ReLu e SeLu).

8. Todas camadas internas como SeLu e 150 épocas: Esta configuração também errou somente 6 exemplares, como o experimento 4, no entanto, o treinamento se mostrou um pouco melhor, uma vez que a acurácia avança mais próxima entre validação e treino. No experimento 4 a acurácia de treinamento converge mais do que a de validação, como mostra a Figura 8, a linha vermelha de validação fica mais abaixo do que a linha vermelha do presente experimento.

```
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Conv2D(32, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(128, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="softmax"),
])
```

Figura 13 (modelo com todas camadas SeLu).

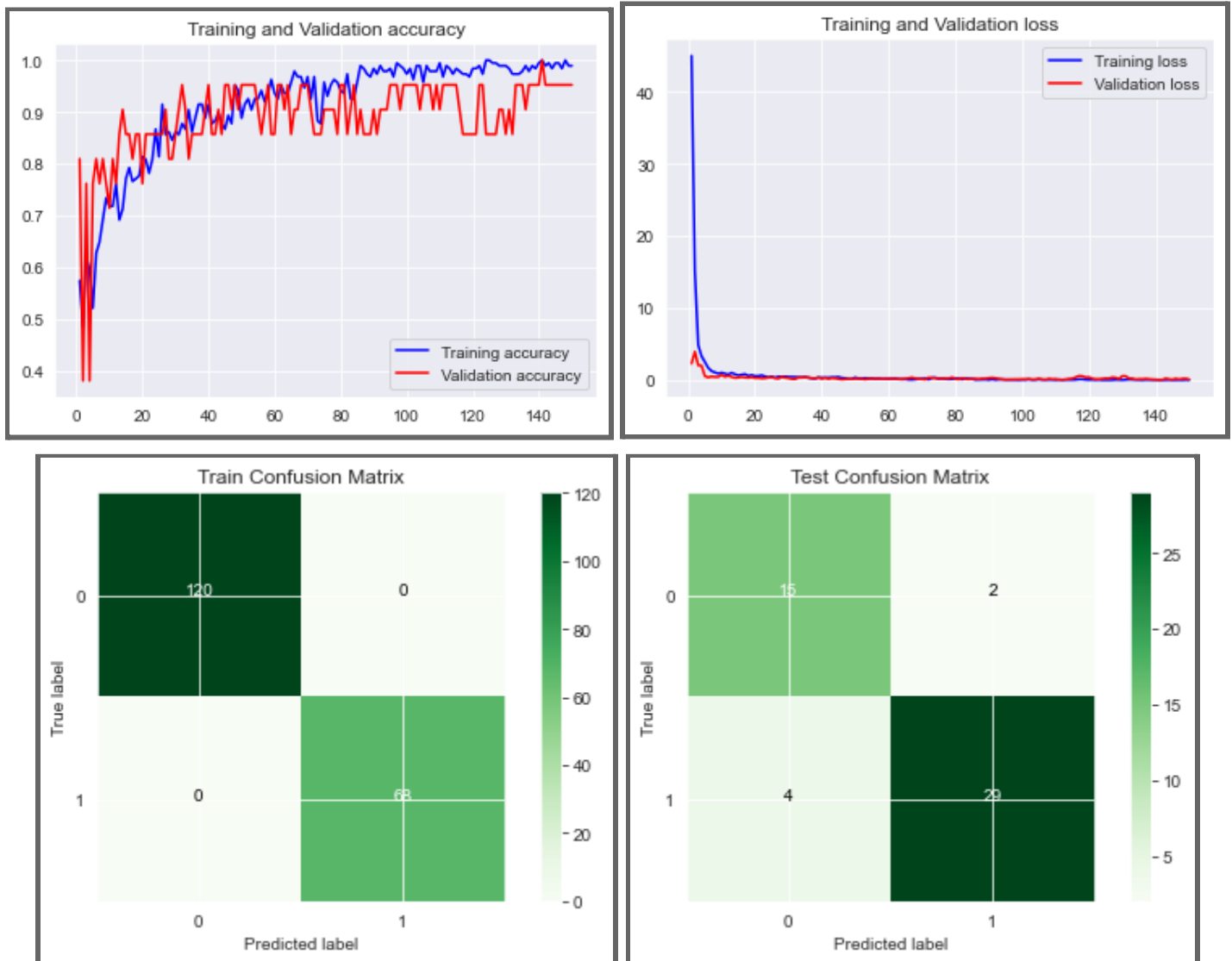


Figura 14 (treinamento com todas camadas SeLu).

9. Camada de saída sigmoid: Na mosca! Utilizando todas camadas escondidas como SeLu e a camada de saída como sigmoid em vez de softmax se obteve um modelo que errou somente 3 exemplares no teste! Acredito que dificilmente teremos um resultado melhor do que esse!

```
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Conv2D(32, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(128, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="selu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="sigmoid"),
])
```

Figura 15 (modelo SeLu e saída Sigmoid).

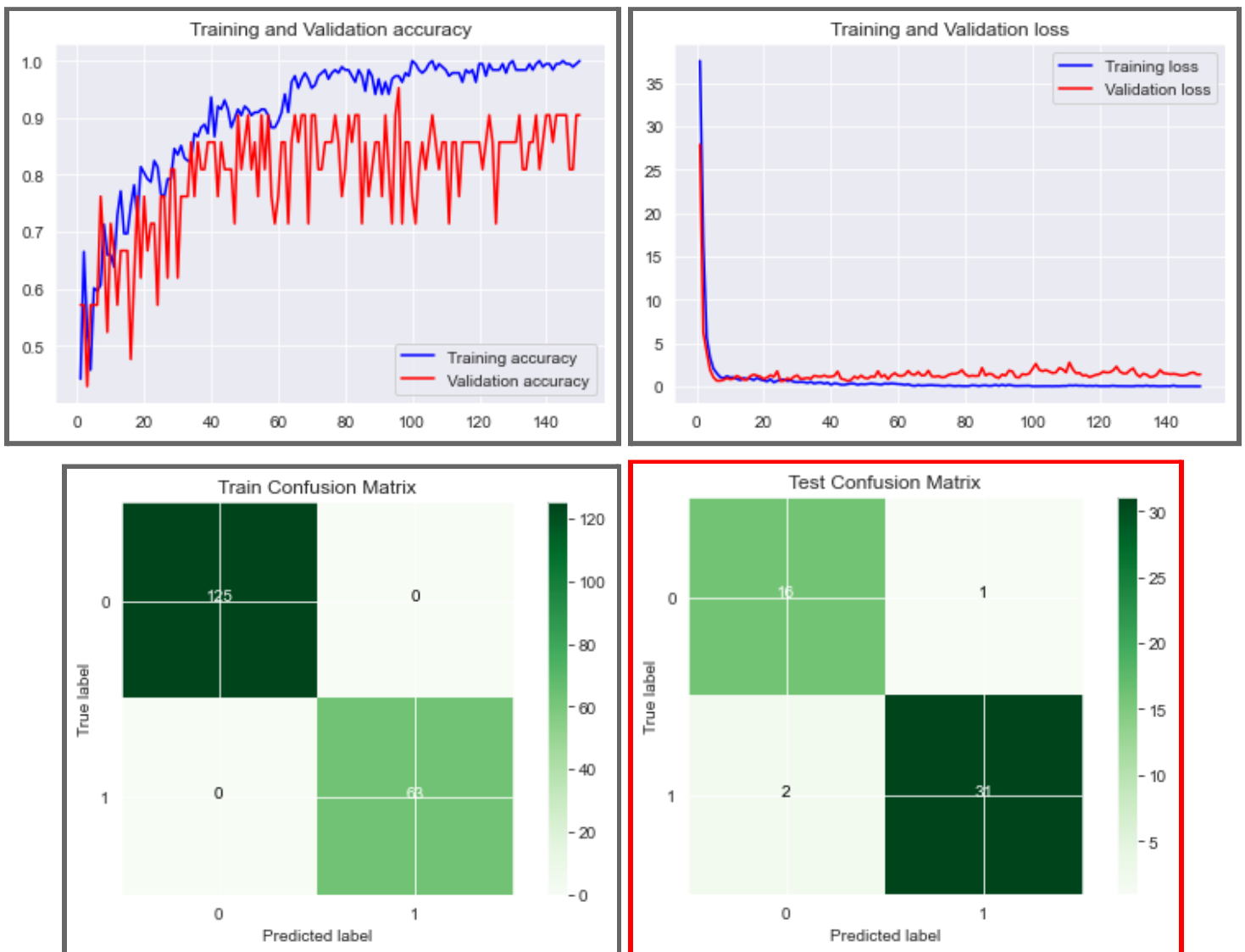


Figura 16 (treinamento vencedor com modelo Selu e saída Sigmoid).