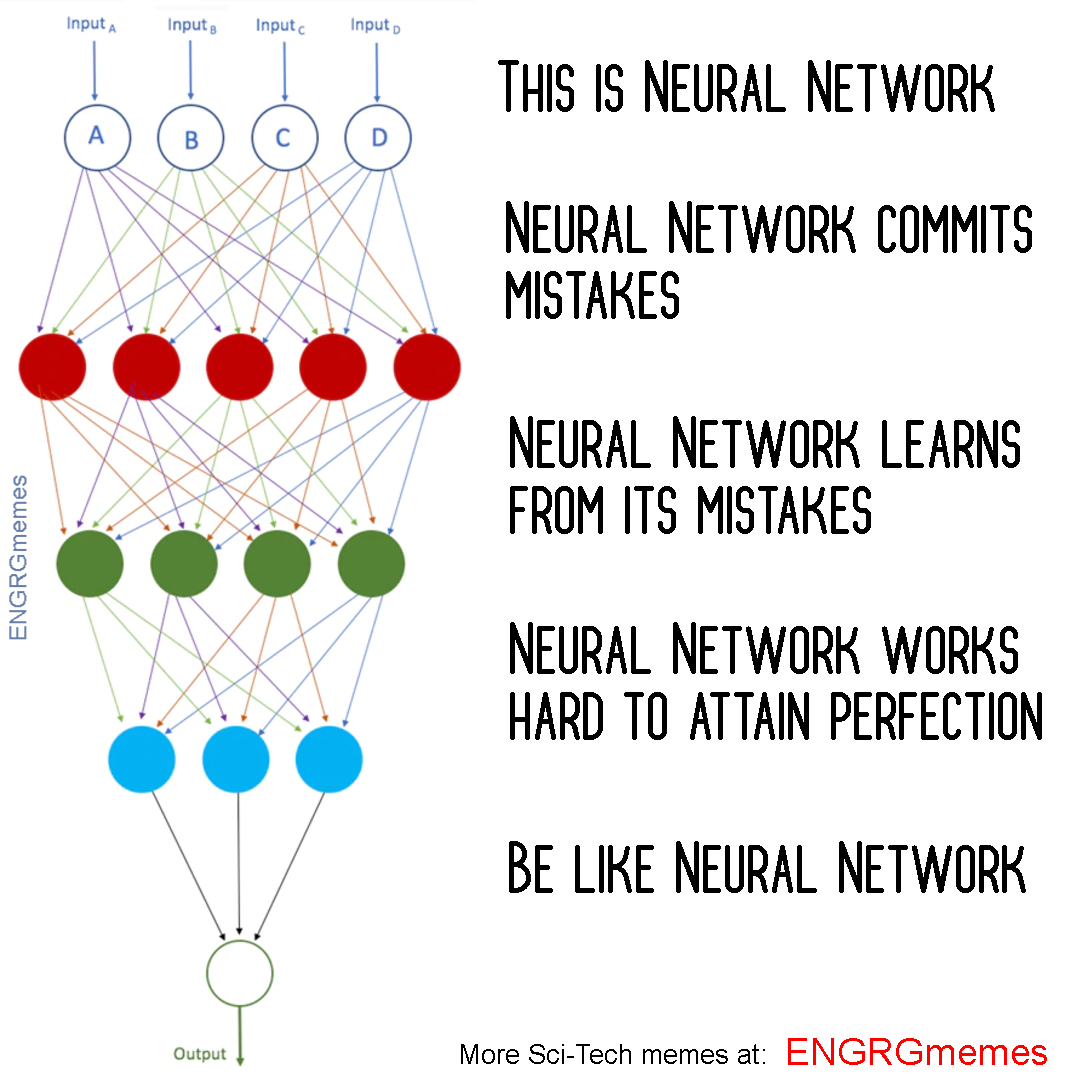
**How are kernels initialized?**

Machine learning works on principle \*\*Make mistakes to correct them\*\*,

* A model performs best if every node learns to identify a different feature, and that happens when each filter has a value which is different from the rest.
* In case all the values are initialized to the same value, the model is not diverse enough to learn different features hence fails to generalize the trends in the data
* First step is we have to initialize our kernel with random values, and calculate the error with the actual value
* Then we make small yet significant changes to minimize the error until we our model generalizes to our data.
* Random Initialization of kernels also makes our model diverse enough to learn different features



**What happens during the training of a DNN?**



Let’s suppose we are training a DNN for image classification, our goal is to classify these images lets say Cats vs Dogs, we have labelled images as our dataset. Now we work on the above principal, **Make mistakes to correct them**

* First, we need to convert our images into the forms our models can understand i.e. array of numbers.
* So these images are then passed into our model, a neural network architecture is shown below. The nodes in hidden layers are also called channels. Each channel as explained the is expert on a specific task
* Initial layers (red ones) would identify edges, gradients, patters, the deep layers would detect parts of objects (eyelashes, nails, whiskers) (green ones) and objects (Face, legs, torso). Hence the channel looking our for that feature, and it it finds it the node gets activated.
* Final layer sums up all the information (activation of previous nodes) collected by the earlier layers and gives a prediction.
* Eg:
  + (2 eyes, nose, 4 legs, whiskers, pointy ears, mean AF --> Cat)
  + (2 eyes, pointy nose, 4 legs, ears, Loving AF --> Dog)
* When initialized with random values, the nodes are bound to make errors, . But then comes the part where we correct these mistakes.
* The error is then propagated backwards into the model to minimize them in small yet significant steps also known as backprop.

**How many times to we need to perform 3x3 convolutions operations to reach close to 1x1 from 199x199 (type each layer output like 199x199 > 197x197...)**

We have to perform the computation 100 times, every time when 3\*3 computlation is performed as explained above,

(199 x 199) > (197 x 197) > (195 x 195) > (193 x 193) > (191 x 191) > (189 x 189) > (187 x 187) > (185 x 185) > (183 x 183) > (181 x 181) > (179 x 179) > (177 x 177) > (175 x 175) > (173 x 173) > (171 x 171) > (169 x 169) > (167 x 167) > (165 x 165) > (163 x 163) > (161 x 161) > (159 x 159) > (157 x 157) > (155 x 155) > (153 x 153) > (151 x 151) > (149 x 149) > (147 x 147) > (145 x 145) > (143 x 143) > (141 x 141) > (139 x 139) > (137 x 137) > (135 x 135) > (133 x 133) > (131 x 131) > (129 x 129) > (127 x 127) > (125 x 125) > (123 x 123) > (121 x 121) > (119 x 119) > (117 x 117) > (115 x 115) > (113 x 113) > (111 x 111) > (109 x 109) > (107 x 107) > (105 x 105) > (103 x 103) > (101 x 101) > (99 x 99) > (97 x 97) > (95 x 95) > (93 x 93) > (91 x 91) > (89 x 89) > (87 x 87) > (85 x 85) > (83 x 83) > (81 x 81) > (79 x 79) > (77 x 77) > (75 x 75) > (73 x 73) > (71 x 71) > (69 x 69) > (67 x 67) > (65 x 65) > (63 x 63) > (61 x 61) > (59 x 59) > (57 x 57) > (55 x 55) > (53 x 53) > (51 x 51) > (49 x 49) > (47 x 47) > (45 x 45) > (43 x 43) > (41 x 41) > (39 x 39) > (37 x 37) > (35 x 35) > (33 x 33) > (31 x 31) > (29 x 29) > (27 x 27) > (25 x 25) > (23 x 23) > (21 x 21) > (19 x 19) > (17 x 17) > (15 x 15) > (13 x 13) > (11 x 11) > (9 x 9) > (7 x 7) > (5 x 5) > (3 x 3) > (1 x 1)