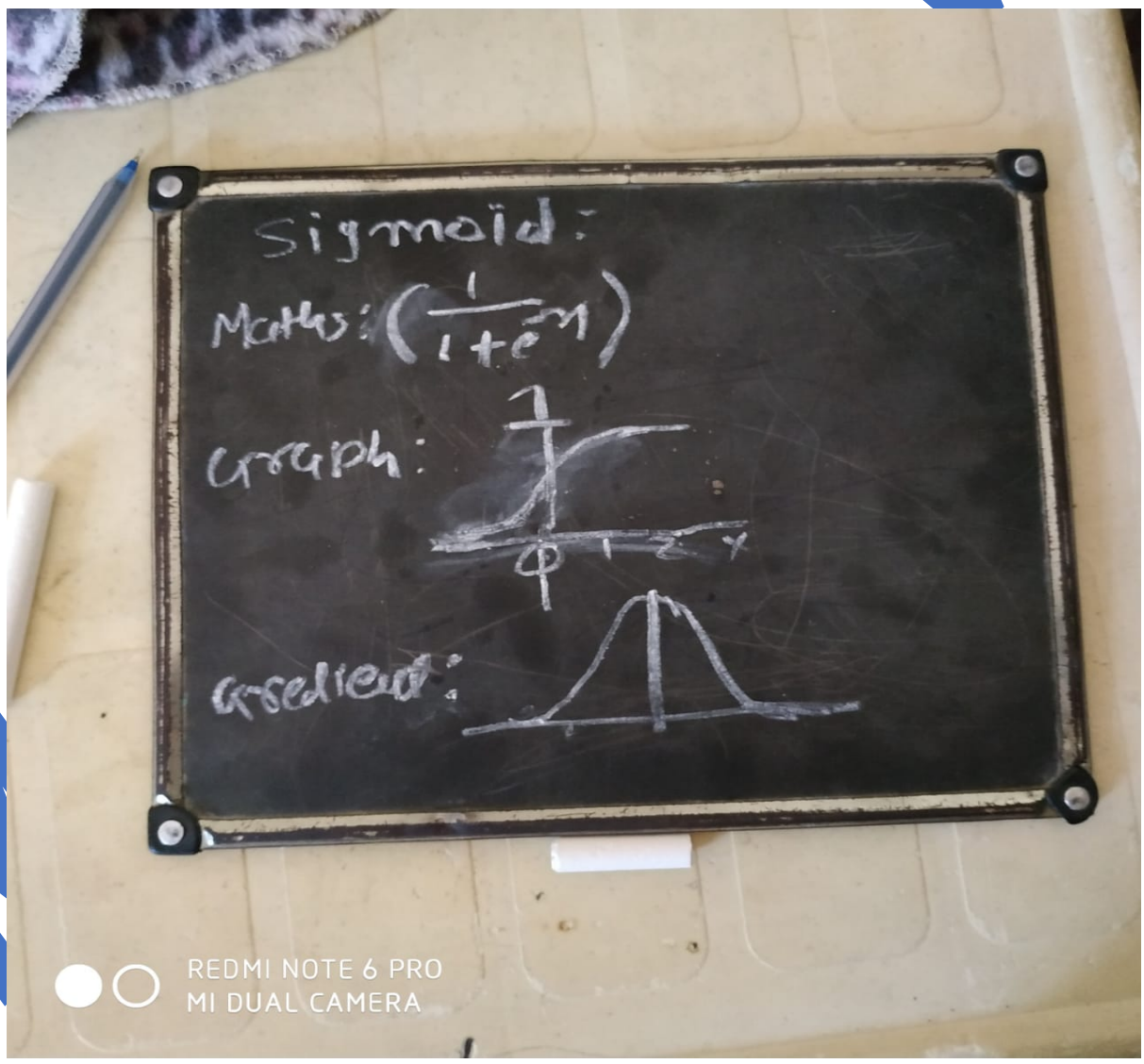
Activation Functions

- Activation functions behave like a gate.
- Activation functions try to normalize the dataset.
- Activation functions always try to non-linearization into a dataset or it'll try to help you out to understand a non-linear dataset or very complex dataset.
- As of now 50 to 60 activation functions are available.

Sigmoid Activation Function

Sigmoid is commonly used activation function, it always gives you smooth gradient. There is a misunderstanding in people that the Sigmoid is a probability function but in real sigmoid is not a probability function. Bellow is the mathematical representation of Sigmoid function, graph and Gradient.



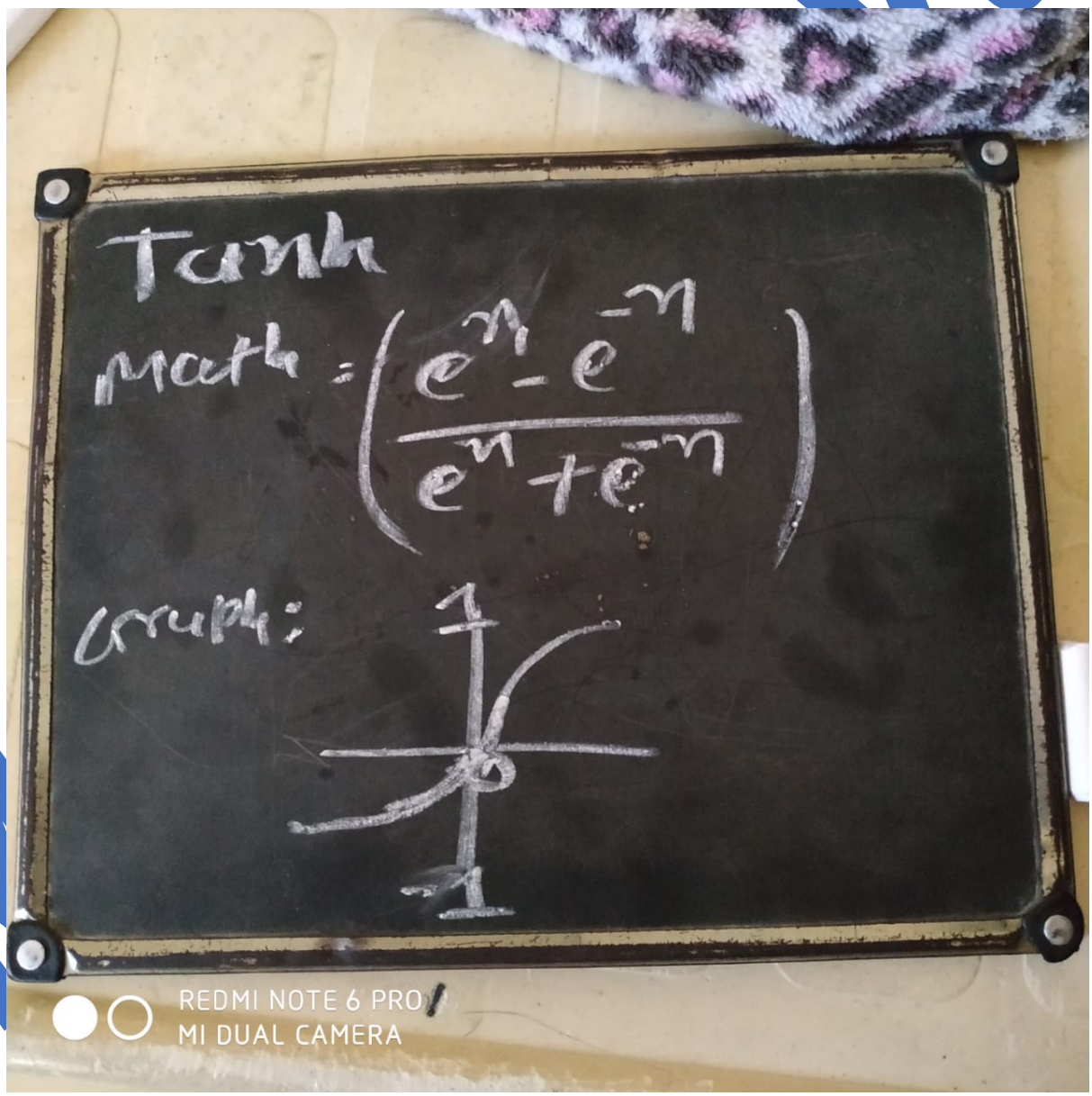
Although, Sigmoid is very widely used activation function, it has many drawbacks.

- Exponential in operation.
- Vanishing Gradient.
- Non-zero centric by output.

Tanh (Hyperbolic Tangent) Activation Function

- Tanh function also known Hyperbolic Tangent.
- Tanh is Zero centric function.
- For normalization of the data it's a good one.
- It can solve a problem in better way compare to Sigmoid.
- It's mostly used in binary class classification and advisable to use it in hidden layer.

Bellow is the mathematical representation of Tanh activation function, graph and Gradient.



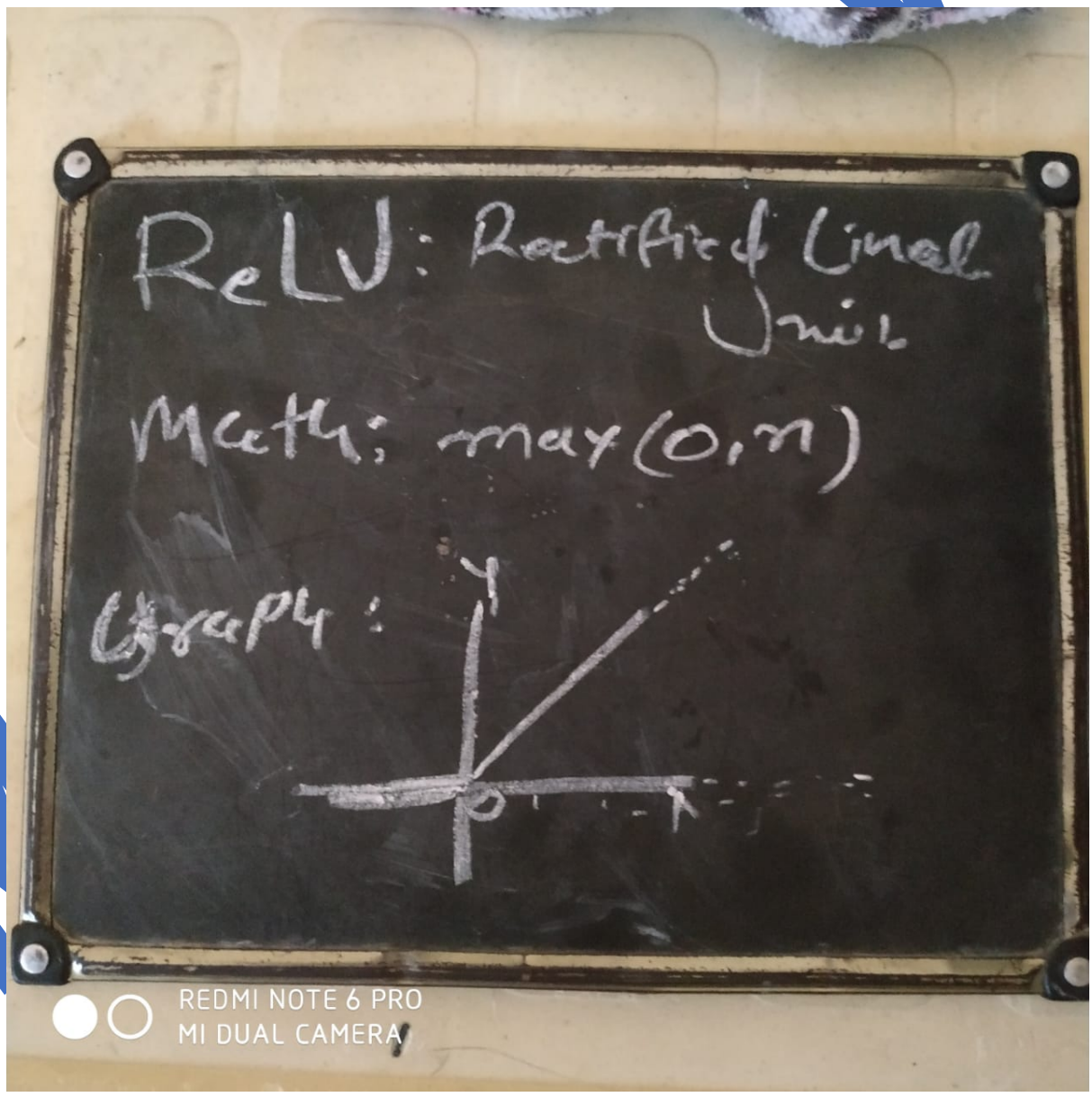
Drawbacks:

- Not very useful in very huge dataset.
- Vanishing Gradient.
- Exponential in operation.

ReLU (Rectified Linear Unit) Activation Function

As we seen, there are a lot of vanishing gradient problem with Sigmoid and Tanh activation function so people have introduce a ReLU function.

- So with ReLU, there wont be any vanishing grading problem on positive input.
- If in backward propogation, ReLU is contributing then it's going to change the weight in a cmatrix way.
- When input is positive then no gradient saturation.
- Calculation speed faster in compare to Sigmoid or Tanh.



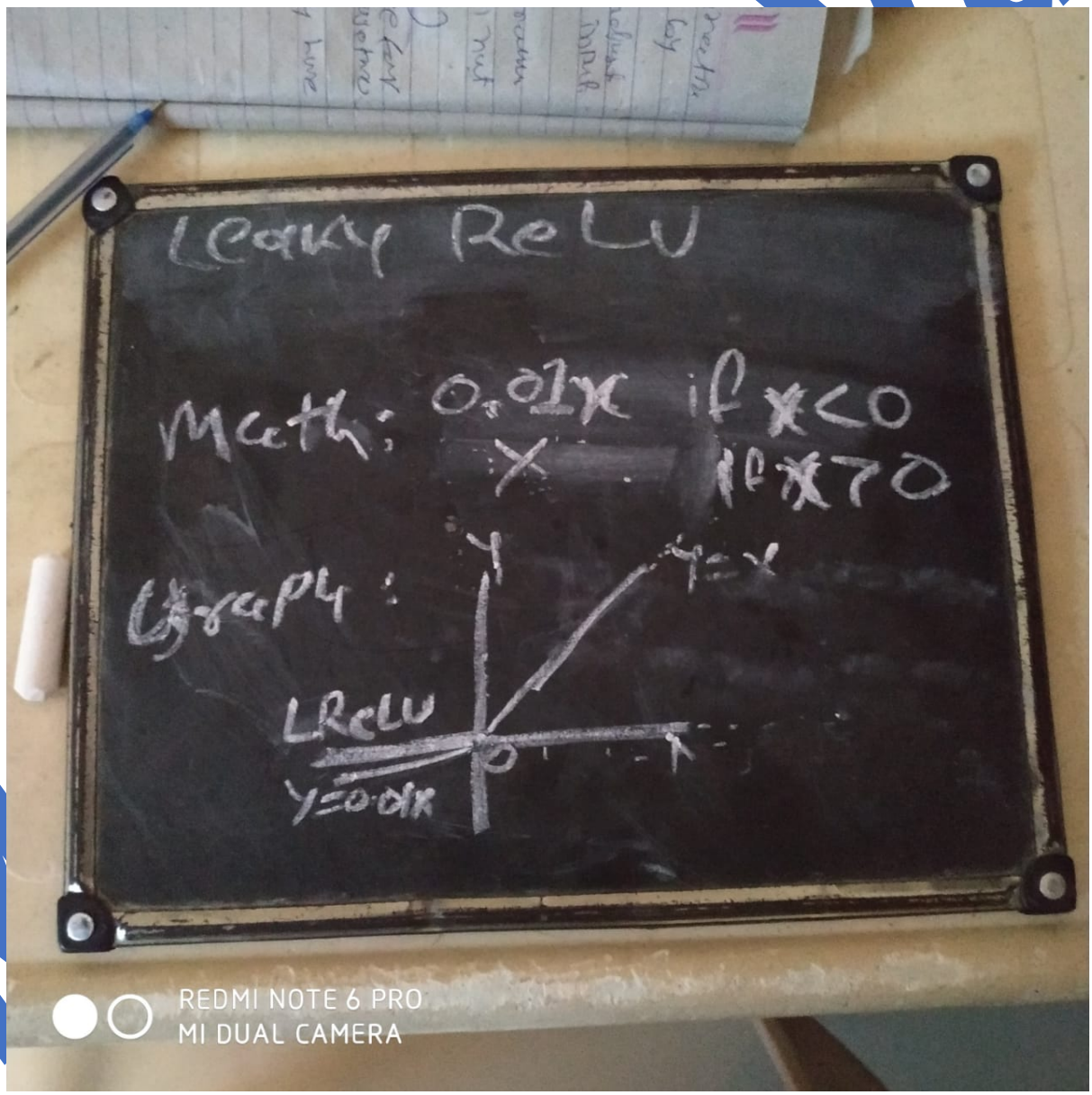
Drawbacks:

- In case of negative inputs, it'll always gives a zero.
- It's not a zero centric function (normalization).

Leaky ReLU (Leaky Rectified Linear Unit) Activation Function

Leaky ReLU has a small slope for negative values instead of all together zeros. For example, Leaky ReLU may have y=0.01x when x < 0.

- It's a simple in calculation and speed faster in compare to Sigmoid or Tanh.
- For -ve, it is adjusting gradient so gradient clipping will not happen.

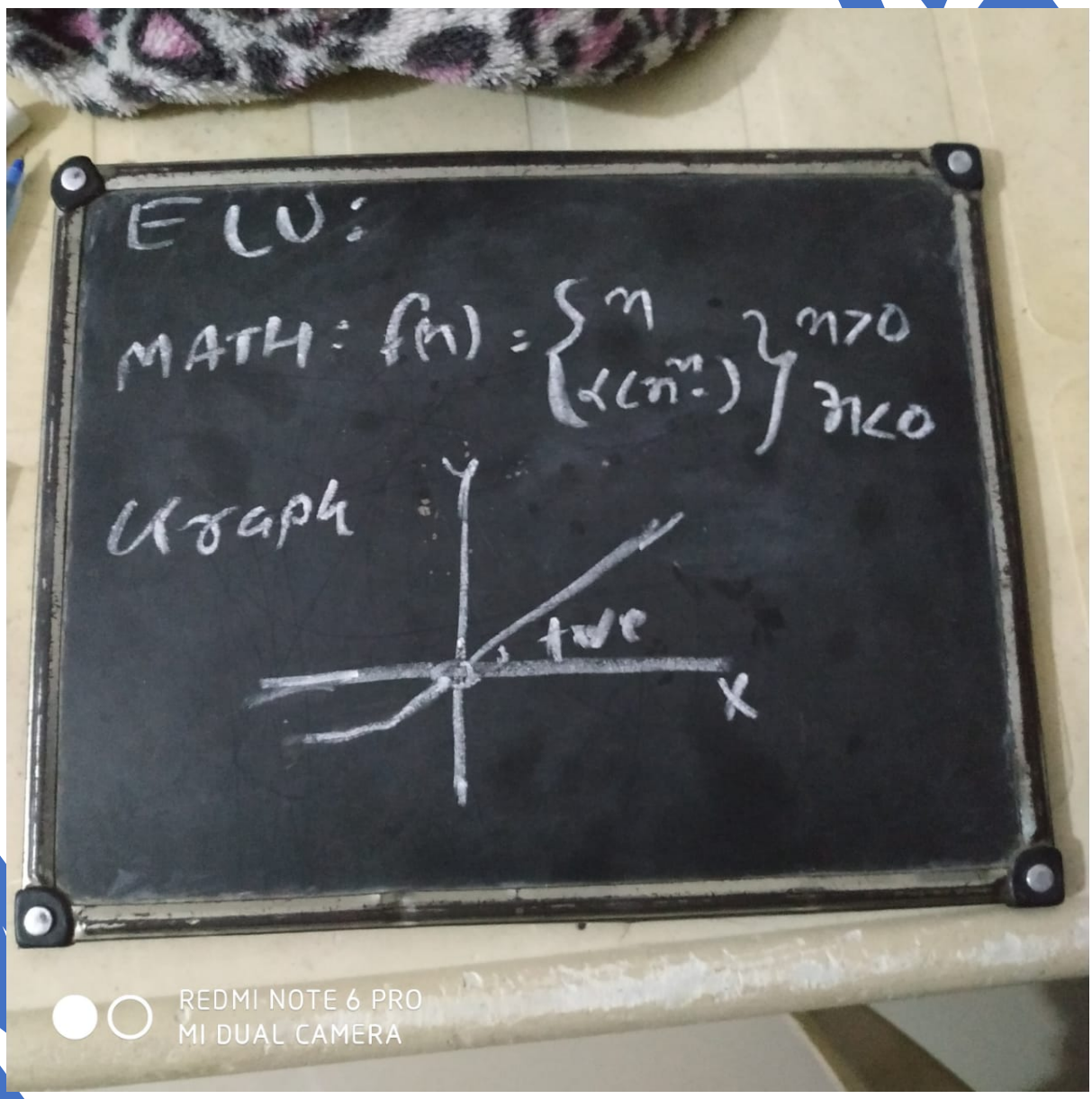


Drawbacks:

- Gradient is not adjusted much for -ve input.

ELU (Exponential Linear Unit) Activation Function

When you use ELU, you will be able to find out that there is no vanishing gradient or gradient clipping problem going to be observed. Although theoretically ELU is better than ReLU but there is no such evidence found ELU is better than ReLU.
ELU is not very commonly used function
The mean or average of output of ELU is zero or closed to zero, so that being said ELU is zero centric function.

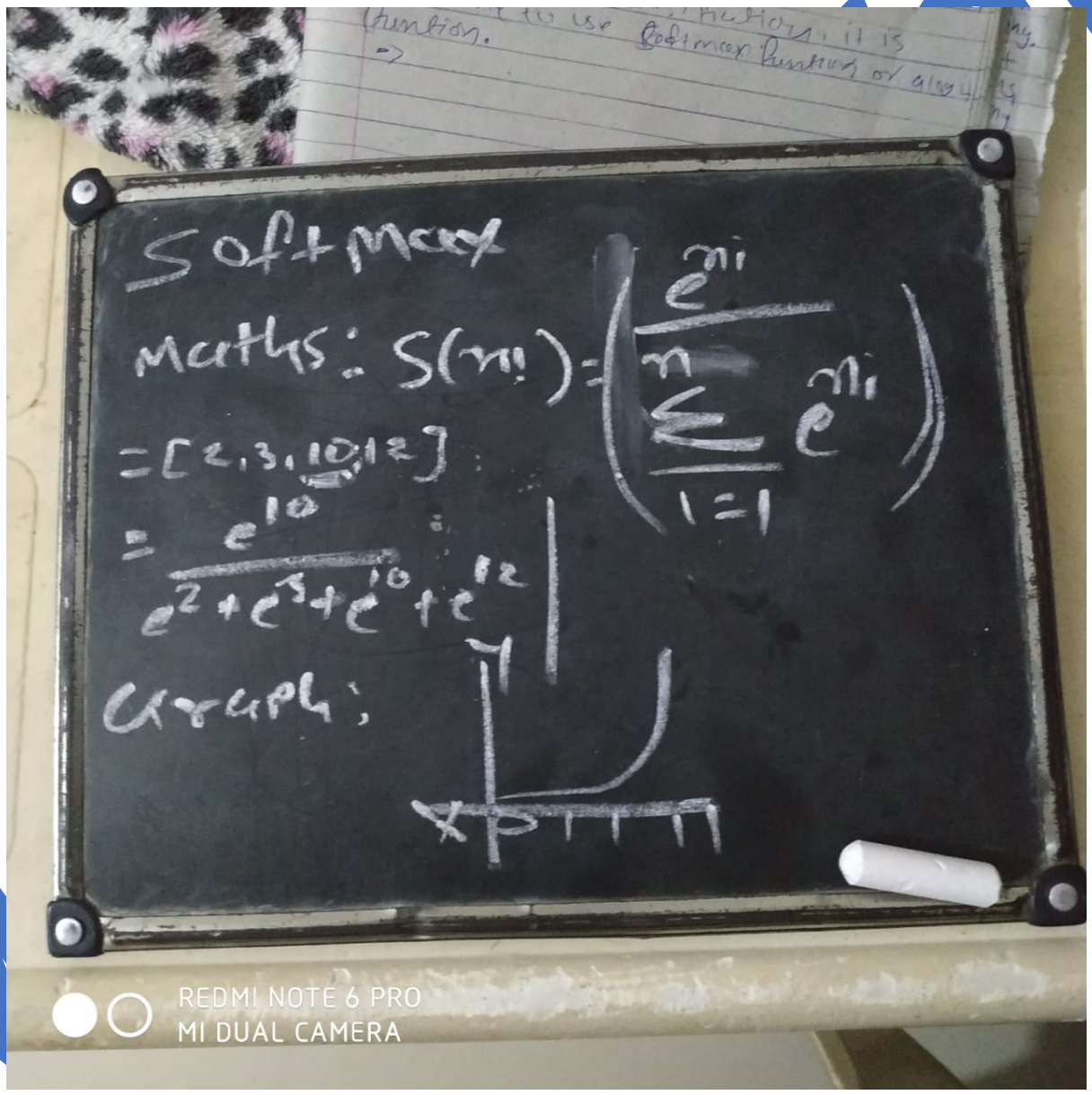


Drawbacks:

- For -ve it's little bit compute expensive.

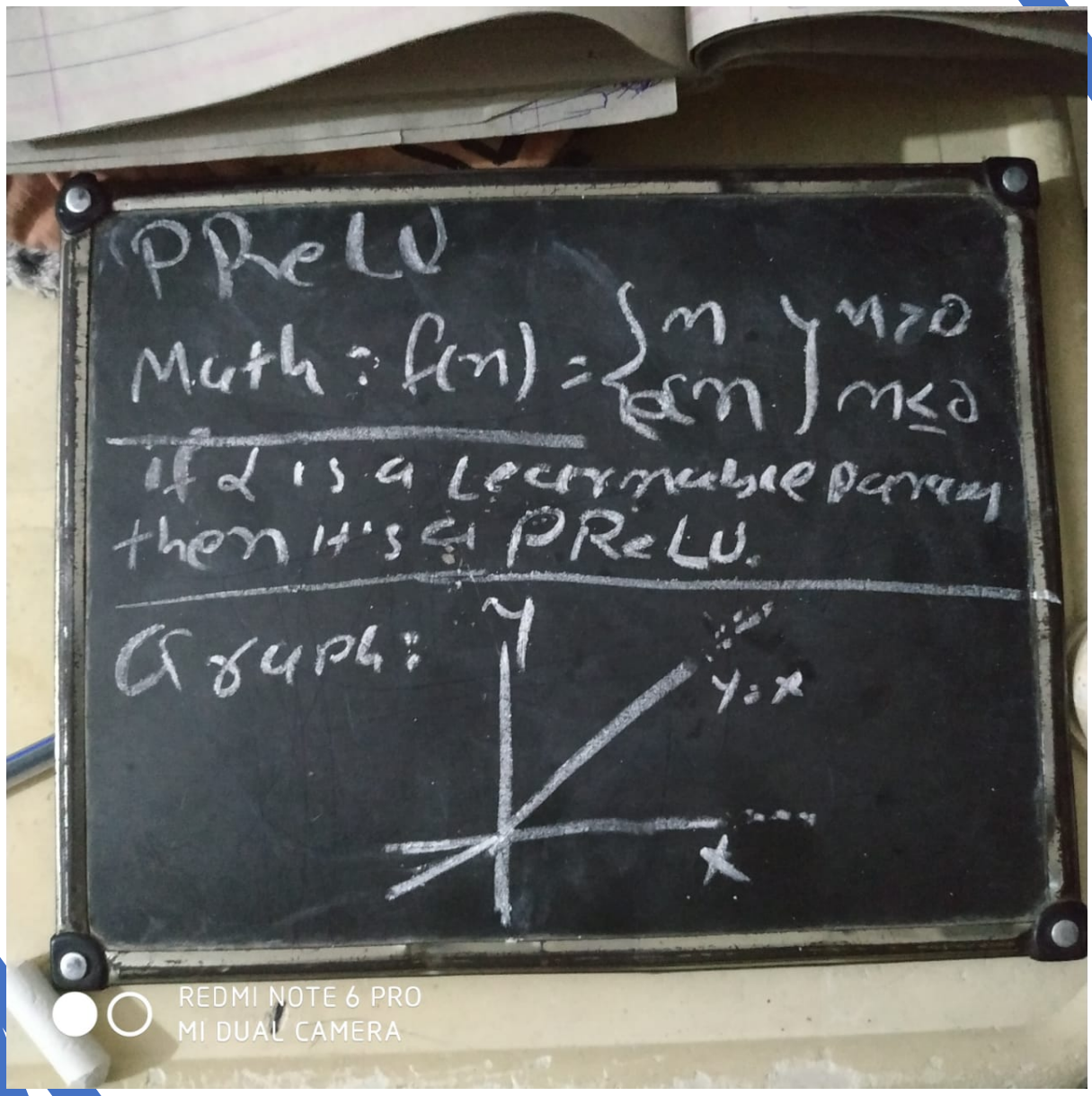
Softmax Activation Function

Softmax always tries to give you the probability of occurrence of a number.
Softmax basically use into an output layer where you have to find a probability of multiple output.
In the multiclass classification it is advisable to use Softmax or Logit function.



PReLU(Parametric Rectified Linear Unit) Activation Function

Parameetric ReLU (PReLU) is a type of leaky ReLU that, instead of having predetermined slope like 0.01, makes it a parameter for the neural network to figure out itself: y=ax when x < 0.



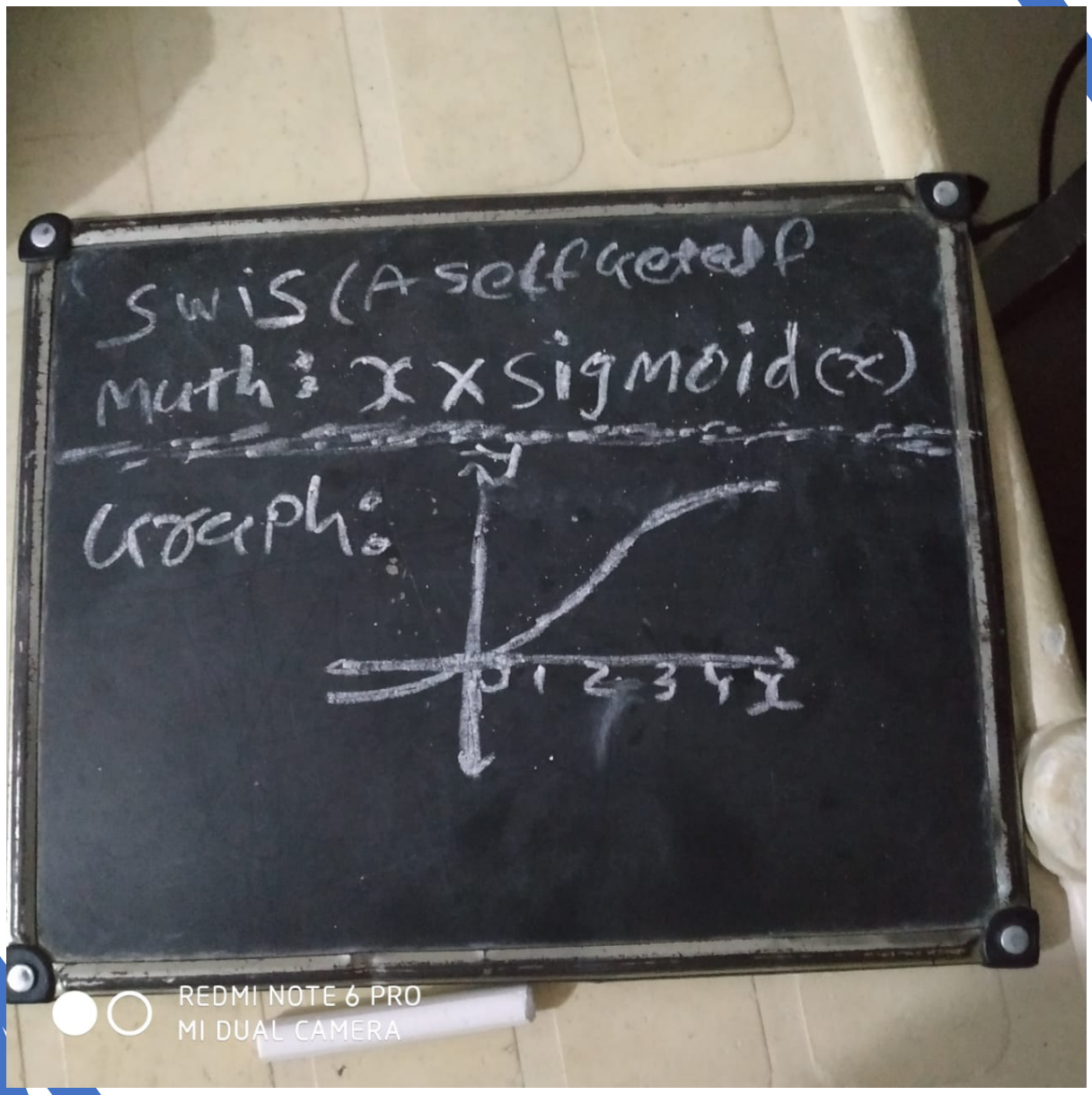
Swish Activation Function

Swish Activation function is proposed by google brain team. Which mathematical representation is f(y) = x * sigmoid (x). Google brain

experiments shows that Swish tends to work better than ReLU on deeper models across a number of challenging datasets.

- Swish Activation will never have vanishing or gradient clipping problem.
- Derivatives (Smooth gradient) will be always there.
- Weight will be get changed on back propagation.

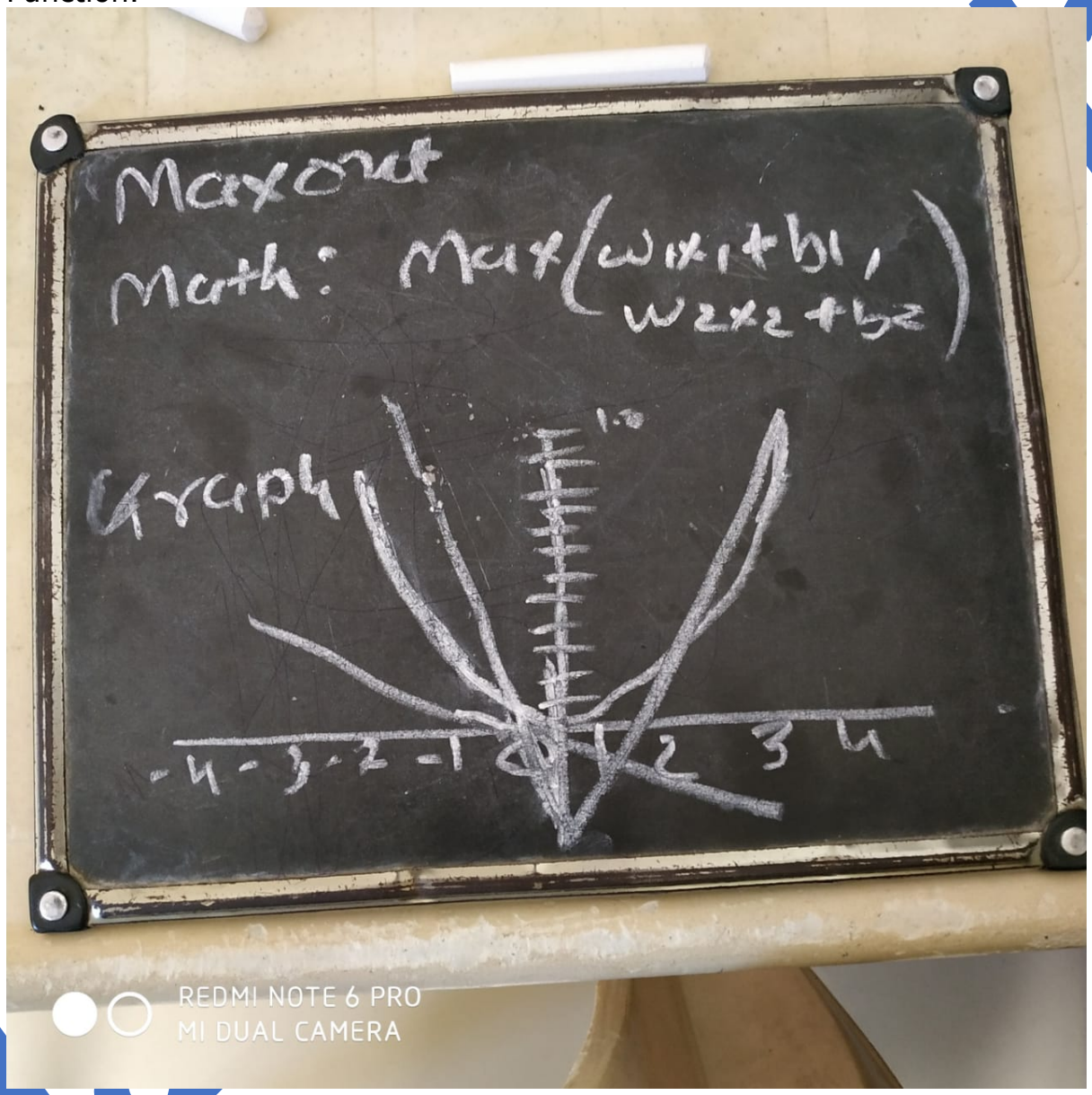
Swish is inspired or designed by sigmoid function for getting a LSTM kind of network or highway kind of a network.



Maxout Activation Function

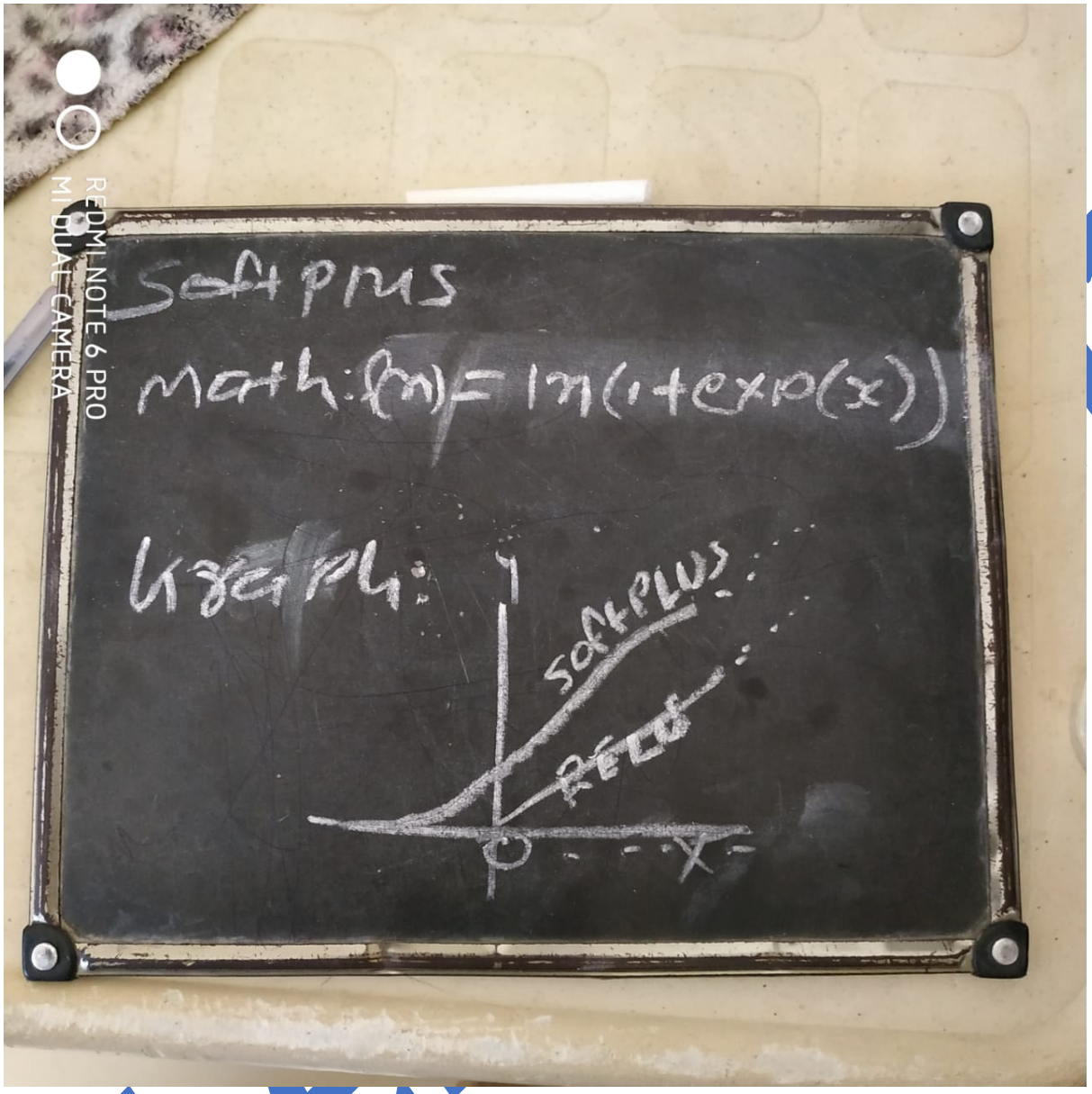
This activation function was invented by Ian Goodfellow in 2013/14 who was inventor of GAN.
Maxout is a generalize version of ReLU and Leaky ReLU.
It is going to give you the maximum of y.

First, it is going to train the weight so, it is going to findout the new value of weight and out of that it is going to give you the maximum.
First of all it is going to learn and out of that learnable parameters, it is going to a final value.
This Activation function doesn't depend on predefined situation.
Bellow is the mathematical replantation of the Maxout Activation Function.



Softplus Activation Function

Softplus activation function is again equivalent to a ReLU function.
Relatively Softmax is a smoother in compare to a ReLU function.
It always accept a range between 0 to infinity.
It does not have vanishing gradient problem.
Bellow is the mathematical replantation of the Softplus Activation Function.

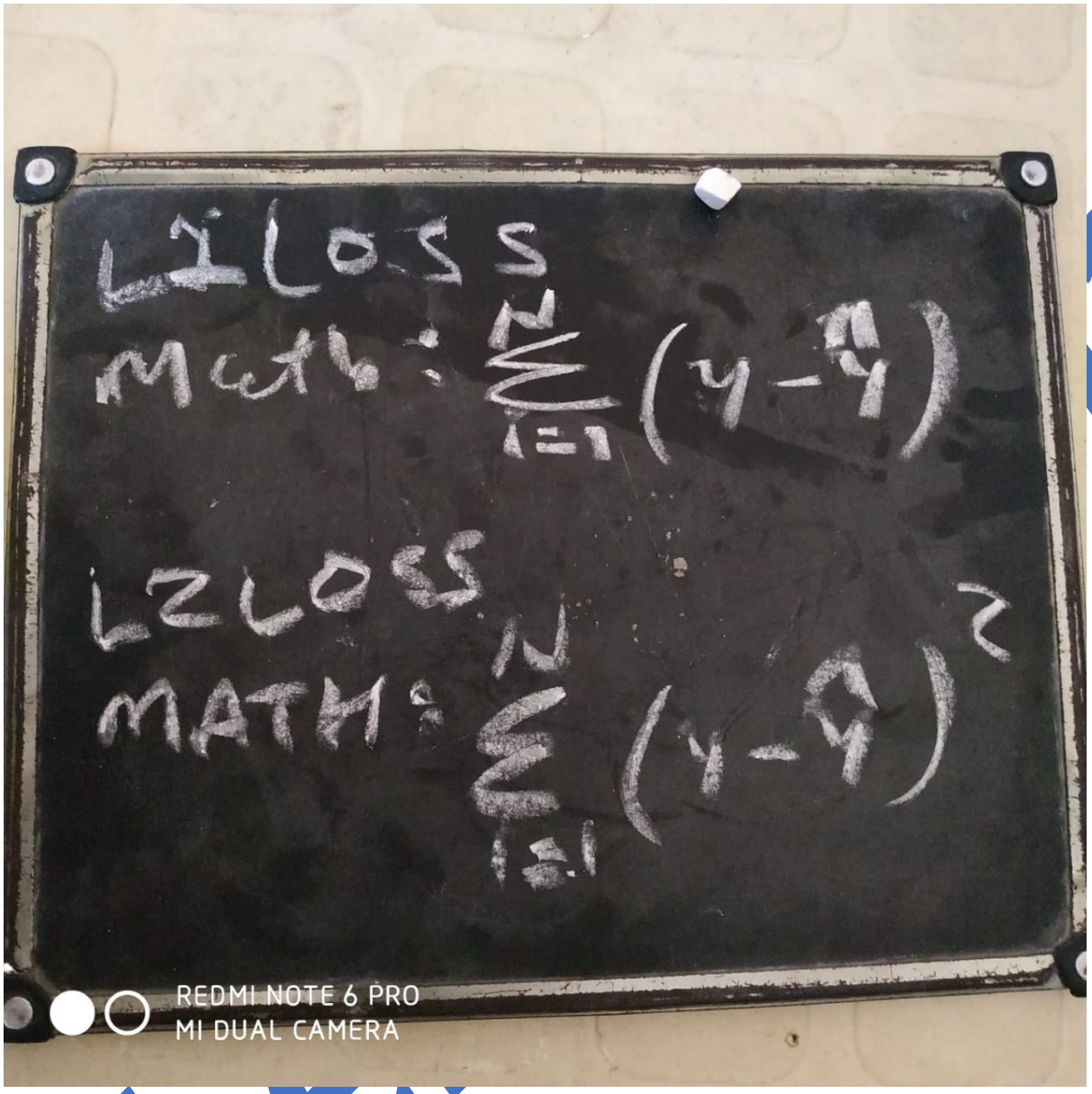

Softplus
Math: f(x) = ln(1+exp(x))
Graph:
SOFTPLUS
RELU
0
X

Cost / Loss Function

As we know that whenever we try to pass data into hidden layer, then there would be Activation function then it'll be pass output layer and there could be activation function that we need to apply. Now whenever you go to get the output $\hat{y}$ then we try to compare it with the actual value ie y for example $y - \hat{y}$.
And based on the comparison you try to calculate the error/loss.
Loss means, what is the error between your expected value and the output.

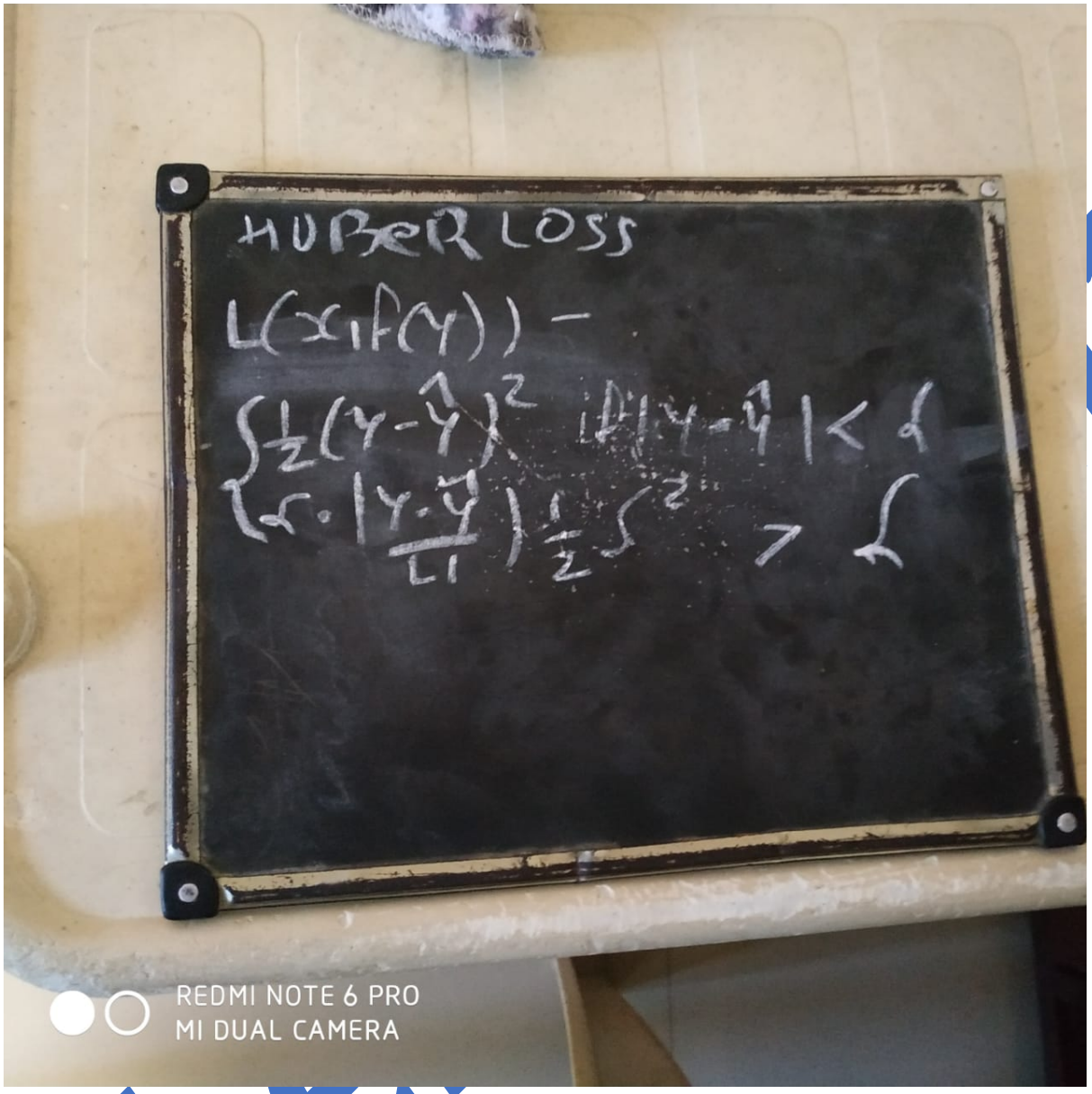
L1 & L2 Loss Function

L1 loss is called as least absolute deviation.
L1 loss tries to minimize error by the absolute value minus predicted value.
L2 loss is called as least squared error.
In case of L2 loss, we always try to find out the squared.
Suppose, there are huge outliers, L2 providing very huge error but in case of L1, that will not happen.
Bellow is the math representation of L1 & L2 loss/error/cost function.



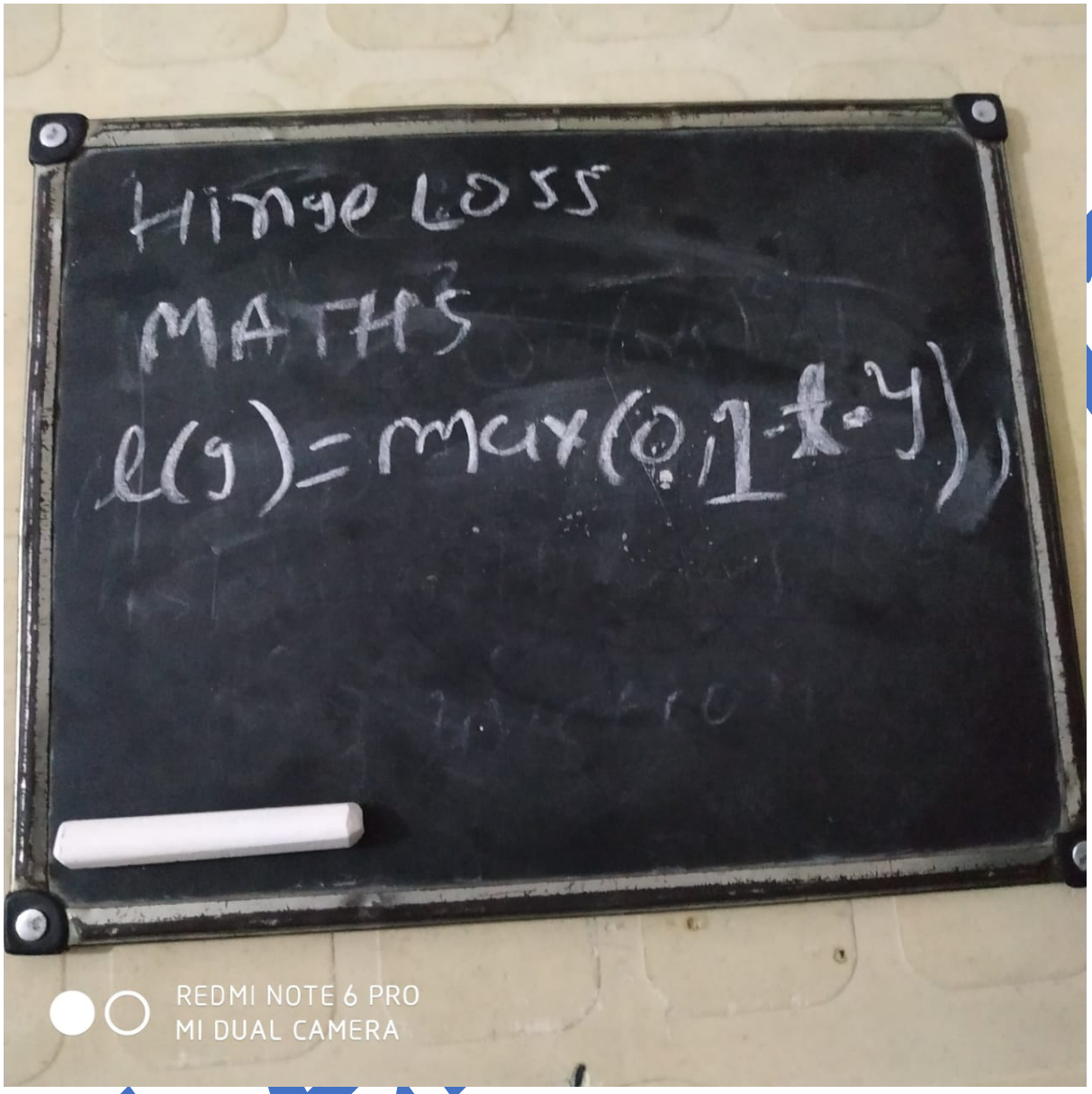
Huber Loss Function

In the L2 loss, if outliers are huge then least squared loss is going to be very high. For example, y – ŷ = 100 then the least squared loss is 10000. To avoid this disadvantage, people has introduced a Huber loss function that trying to control the huge outlier value.
There is a term Delta (δ) which is nothing but outlier value or threshold. So, if absolute value minus predicted value is lesser than Delta (δ) then error will be calculated by **1/2 * (y – ŷ)2**.
And if greater than Delta (δ) then **δ * | y – ŷ| - ½ * δ2.**



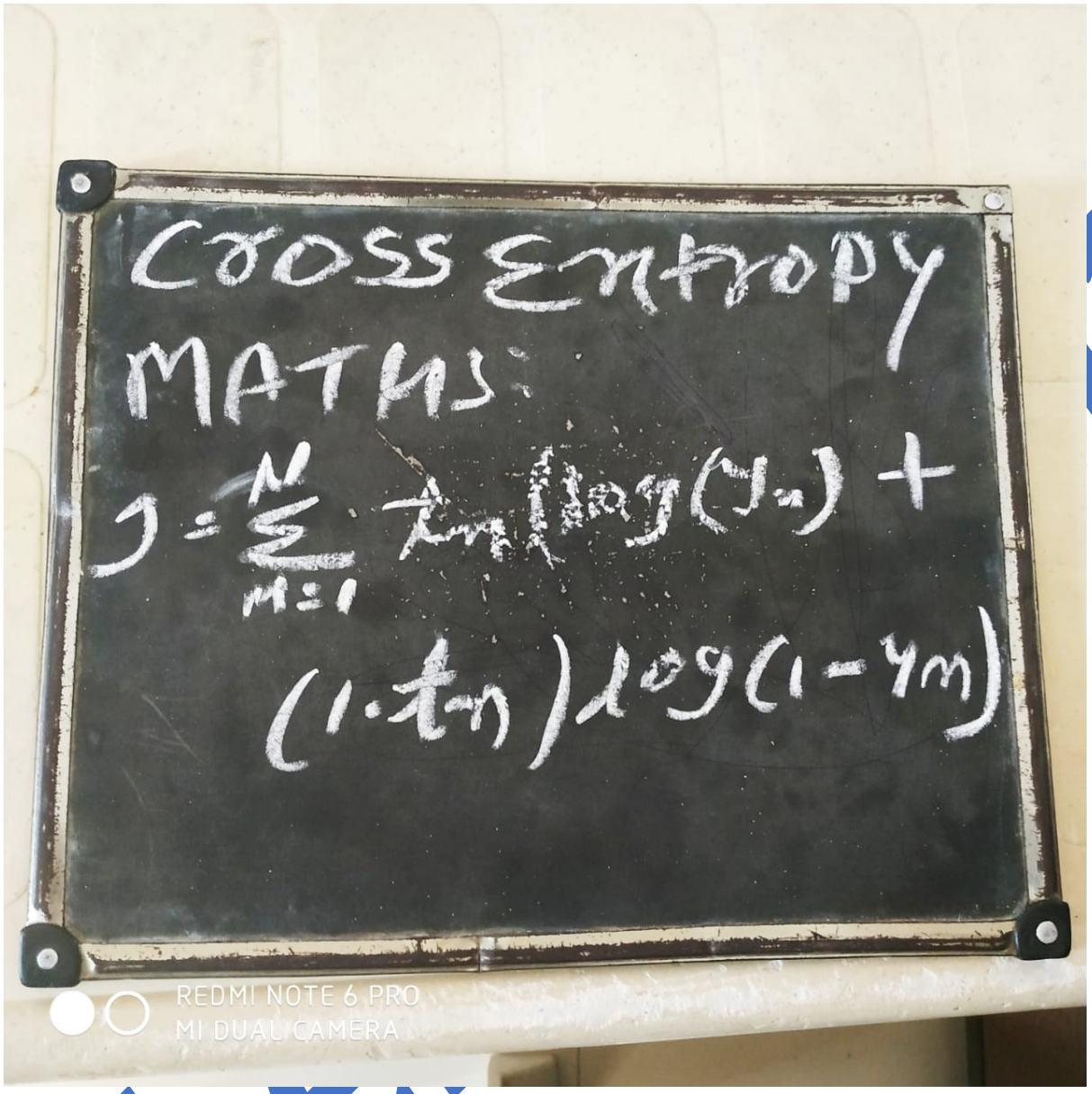
Huber Loss Function

Hinge loss is a function which is been used in the classifier.
Hinge loss is used in the maximum-margin classification, most notably for support vector machines (SVMs).
Formula looks like max (0, 1-t*y), where t is nothing but number of classes.
If it is closer to y is to is small the loss.
Bello is the mathematical representation of Hinge loss function.



Cross Entropy Loss Function

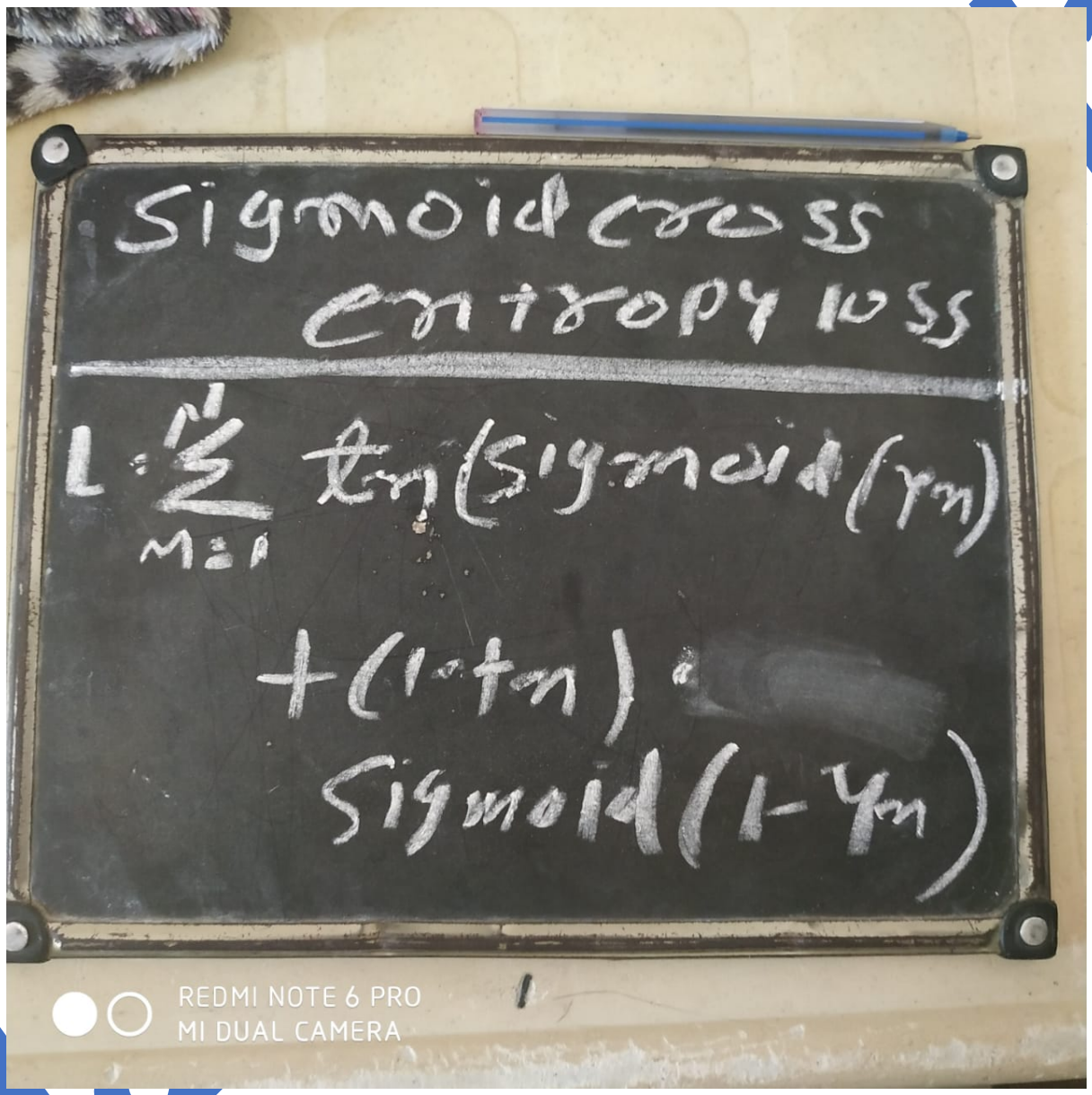
Cross Entropy is very popular and widely used loss function.
Cross Entropy mainly applies into Binary class classification.
In the logistic regression machine learning algorithm this cross entropy function being used internally.



Binary class classification means “t” which could be (0,1) or (-1,1). Here “t” is nothing but the classes and the predictive value is the probability value and the loss is defined in terms of a cross entropy. So the rule, range it is going to provide you will be 0 to 1.

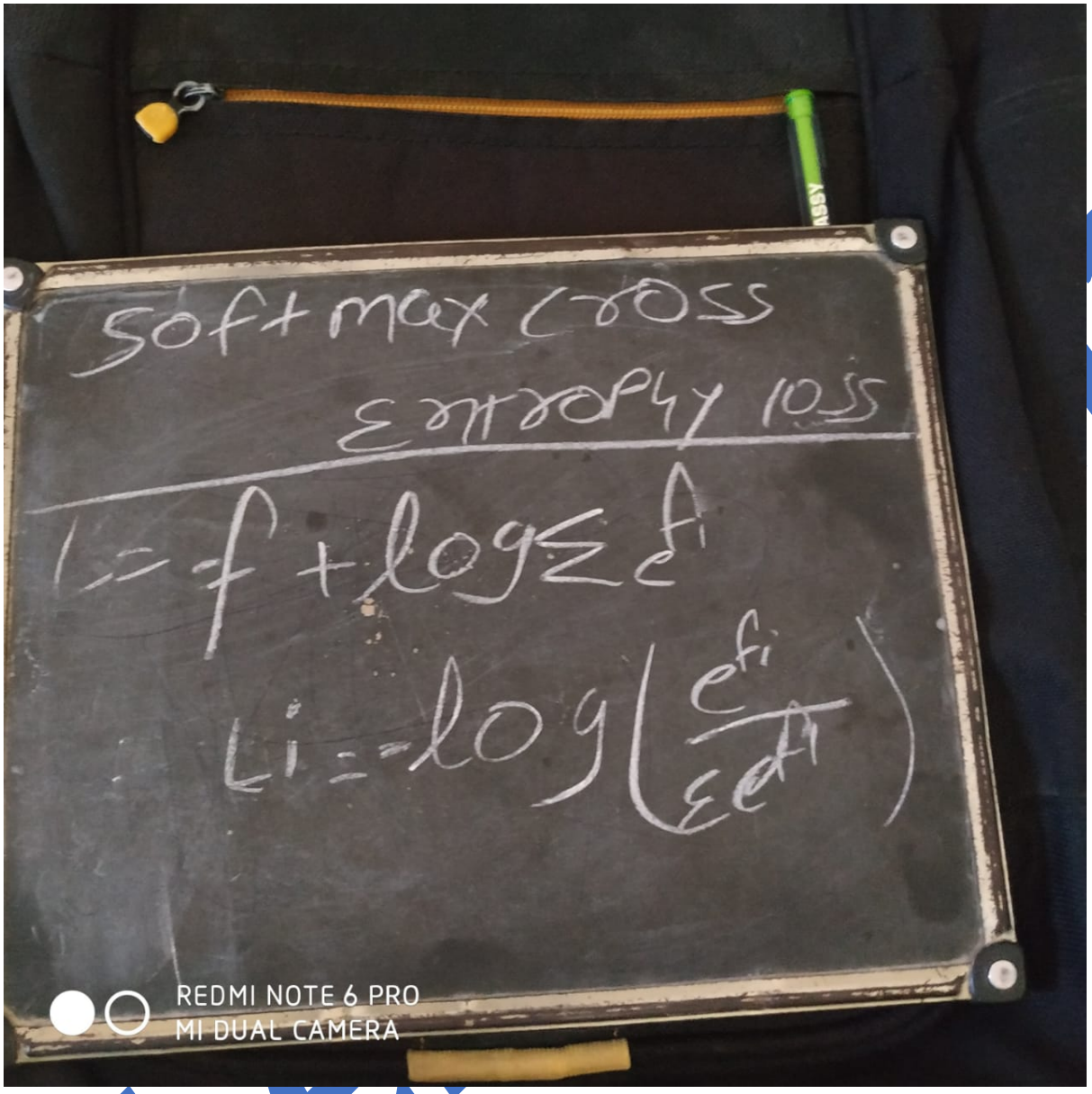
Sigmoid Cross Entropy Loss Function

Basically, in case of cross entropy, you need to predict the predict the possibility so unless and until you get the final possibility, you will not get the loss. But in case of sigmoid cross entropy function, instead of using y, you can use the Sigmoid function and you will be getting your loss.



Softmax Cross Entropy Loss Function

The formula of loss is like Li = (-log (efi/Sigma(e powered of fi)) where (eni/Sigma(e powered of ni)) formula taken from softmax math function, so with respect to softmax cross entropy loss we use same function with respect to log loss.


Softmax cross
entropy loss
L = -f + log Σ e^f
Li = -log (e^fi / Σ e^f)
REDMI NOTE 6 PRO
MI DUAL CAMERA