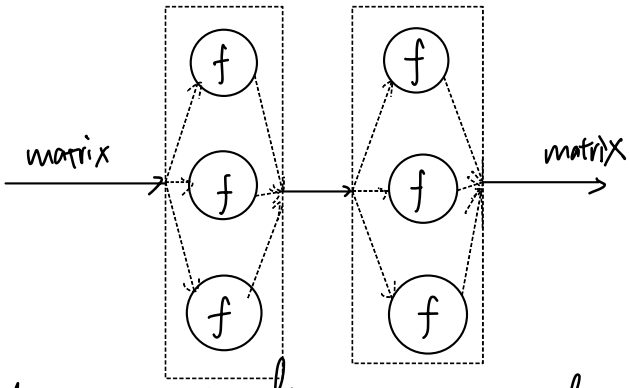
(a) (i)

mapMatrix = mapV . mapV



it applies this function to each element of each row of the matrix

then each column

which means it applies the function to every element

the different rows/ columns can run parallelly

(ii)

reduceMatrix f = reduceV f . mapV (reduceV f)

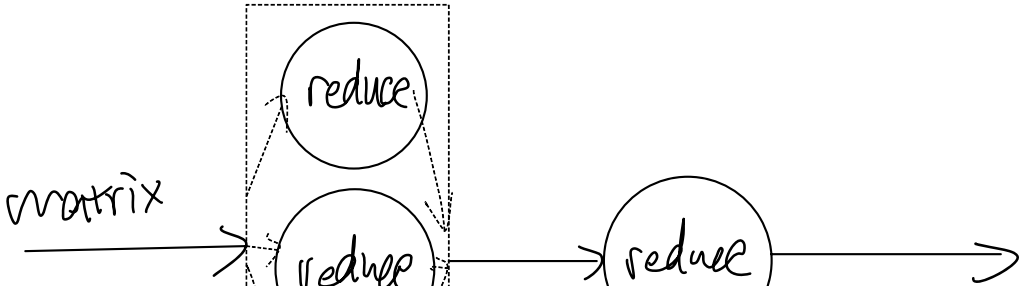
do reduce again

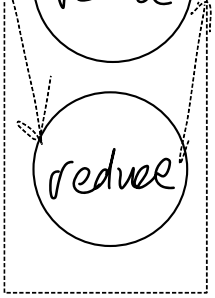
reduce each row into a single value

first reduce each row into a single value,

then apply the reduce function again to the resulting row-wise reduction

the different rows can run parallely





(b) (i)

grayscale = mapMatrix (convert.fromVector). mapV (groupV 3)

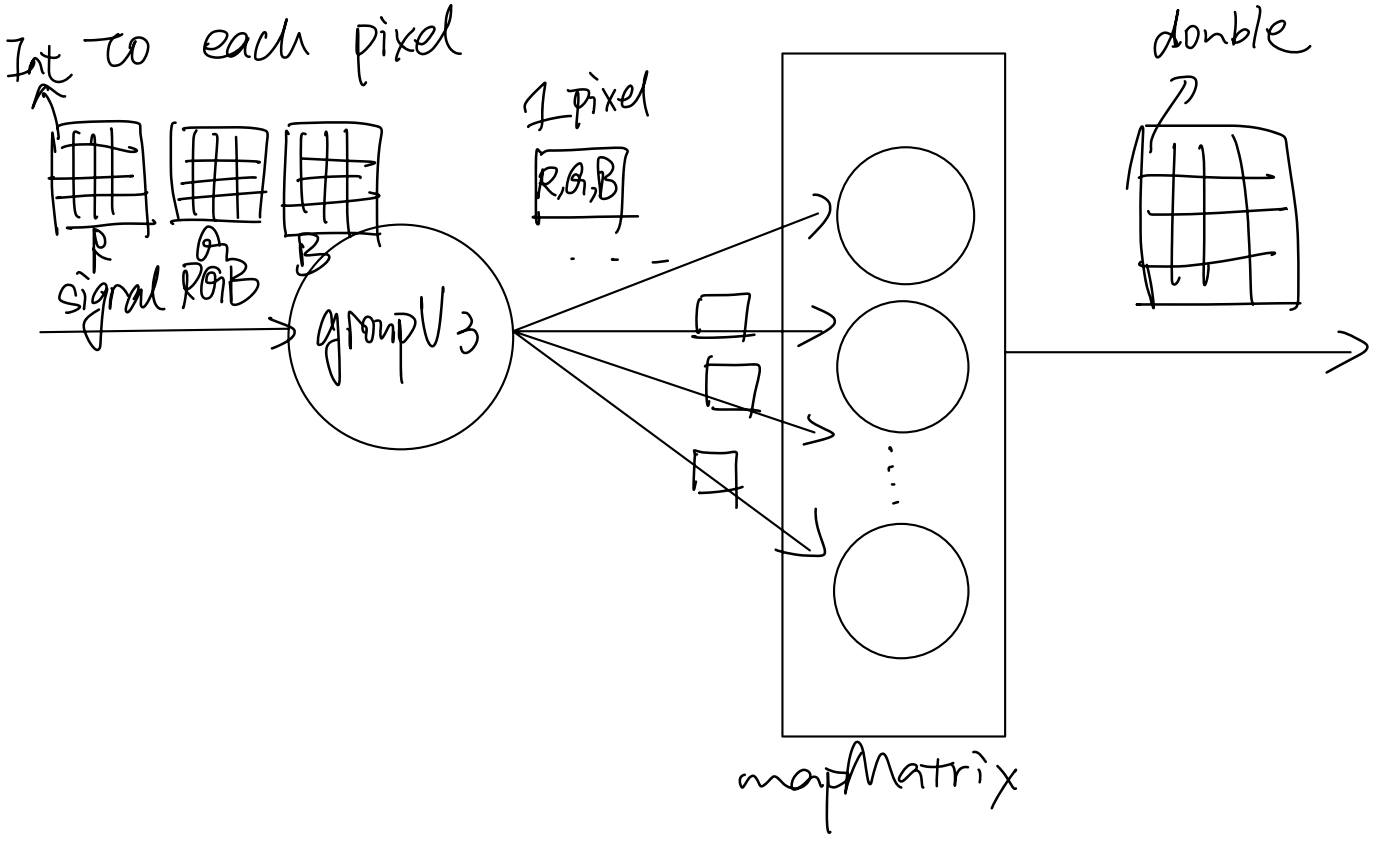
using the function to convert RGB into a grayscale value

converts the pixel vector from its internal representation to a tuple of 3 values

group every 3 elements of each row of the image matrix into a vector.

3 means R, G, B channel

first grouping RGB values into pixel then apply the grayscale conversion to each pixel



(ii) resize :: Image Double → Image Double

resize = mapMatrix (/4) . sumRows . sumCols

divide the image by 4

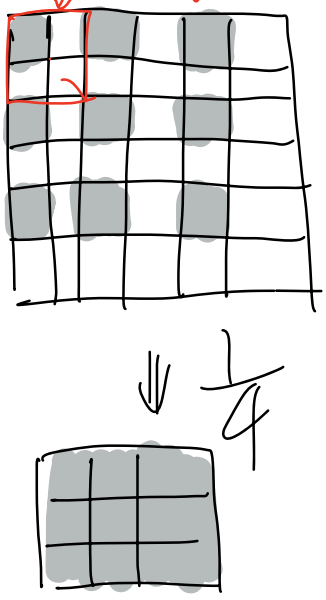
where

sumCols = mapV (mapV (reduceV (+))) . groupV 2

apply reduceV(+) to each pair — Sum up — group every 2 elements of each row into vectors

sumRows = mapV (reduceV (zipWithV (+))) . groupV 2

apply + to each pair of elements



$\Downarrow \frac{1}{4}$

it's a downsampling process.

first horizontally downsampling

the vertically

finally normalize