Project 2 Section 2 – House Pricing

96222031 – Soroush Heidary

Main Idea :

The main idea was to merge two (or more) neural networks, one to takes the pictures as input, the other taking the numerical inputs of the excel file

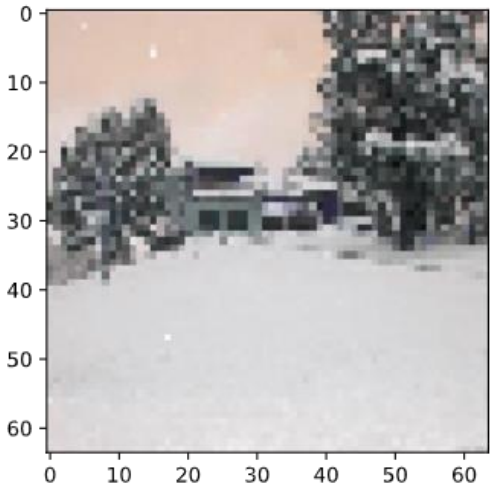
We'll simply play around with different structures to see which works the best

And as for the data, our pictures (just looking at them before doing any work) seem not really usefull to train a network on, and we won't do any augmentations on them, since we don't know which pattern could help the network with the pictures.

Though we will save the RGB channel, because it looks like the color of the house and it's environment could be the most important feature the CNN would want to have

We'll only reduce the picure shapes to match (64, 64, 3) mostly because of the computational power we have (and the different shapes of of pictures)

After the reduction the 16th picture would look like this :



(I know the colors changed :\, but since all the pics are getting changed the relativity still remains the same, so we should be fine)

Data pre processing :

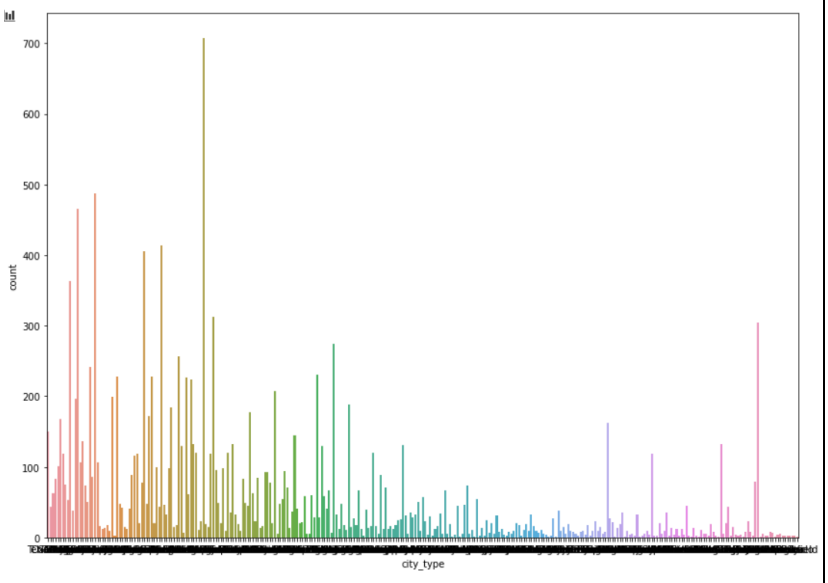
Let's first take a peak at what our numerical data looks like

In the data we have a loot of unique values for street and city columns, but as shown in the pictures, there are some values happening more frequently than others

image_id	street	city	n_city	bed	bath	sqft	price	alt_bath	street_type	city_type	plate
15023	0 H Street	Oxnard, CA	261	3	2	1653	575000	1	Street	Other	0
18	0 Oak Creek rd	Tehachapi, CA	367	4	1	1980	1995000	1	Other	Other	0
11768	0000 Roadrunner Ridge	Valley Center, CA	384	2	1	1045	460000	0	Other	Other	10000
3263	1 Anapamu Street	Ladera Ranch, CA	177	4	4	3700	1595000	1	Street	Other	1
3055	1 Flintridge Avenue	Ladera Ranch, CA	177	6	4	3100	1075000	0	Avenue	Other	1
...	...	...	...	...	...	...	...	...	...	...	...
13174	Address not provided	Carlsbad, CA	67	3	2	1999	805990	1	Other	Other	10000
15207	Address not provided	Camarillo, CA	59	4	3	2200	709000	0	Other	Other	10000
13526	Address not provided	Atascadero, CA	21	3	3	2000	1675000	0	Other	Other	10000
5197	Residence Club Cove	La Quinta, CA	175	3	3	3567	1600000	1	Other	Quinta	10000
7711	Vista Calico	La Quinta, CA	175	3	3	2924	850000	0	Other	Quinta	10000

Lets take a look at how many unique values are in those 2 columns :

This picture shows the distribution of each value in city column :



Well obviously, not even countable by eyes, but some are happening more frequently, so we'll try to set a threshold, and all the other who have less than that value_count will be labeled as 'other'

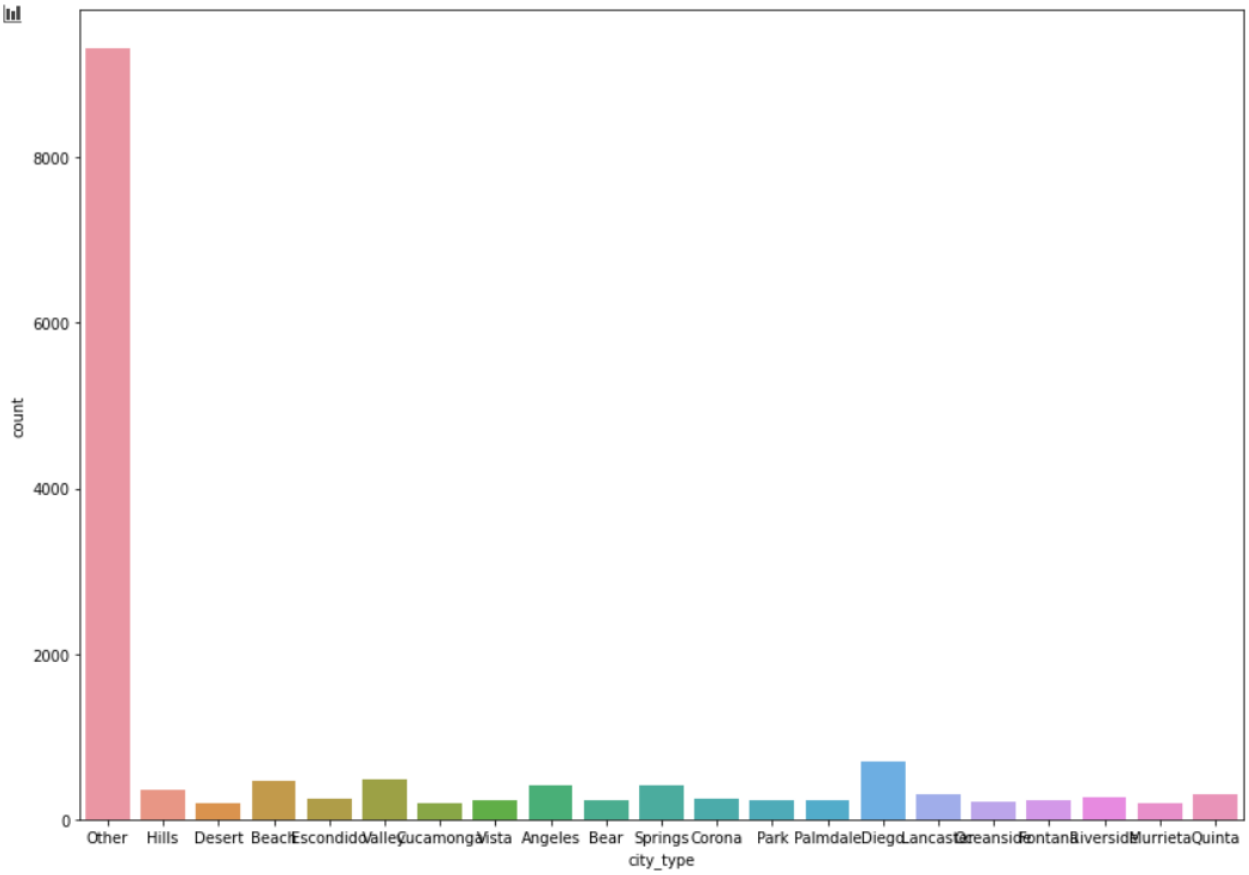
(for the street column, there were around 12000 column and the

demonstration like this one wouldn't be possible, but the same procedure has been applied to street too

This is what the city column unique values looked like, it may seem that what we did was useless but after I tried both, it made the loss rate a little better

And the idea was that the type of the habitant would affect the price and it does seem logical, like those who have a house above a hill, 'hill label' would probably have higher prices for their house,

Another really good approach was to make an Google Maps API and find the streets and cities who lay next to each other and do a clustering algorithm(I read this in a Kaggle notebook, but didn't have the 'hosele' to implement it :)



(same with the street_column)

Training Of Out Network :

For this part we'll try a couple of approaches including 2 main sections

Regression models, and classification models

For the regression part, it's as easy as it sounds, we have an output shape of (1)

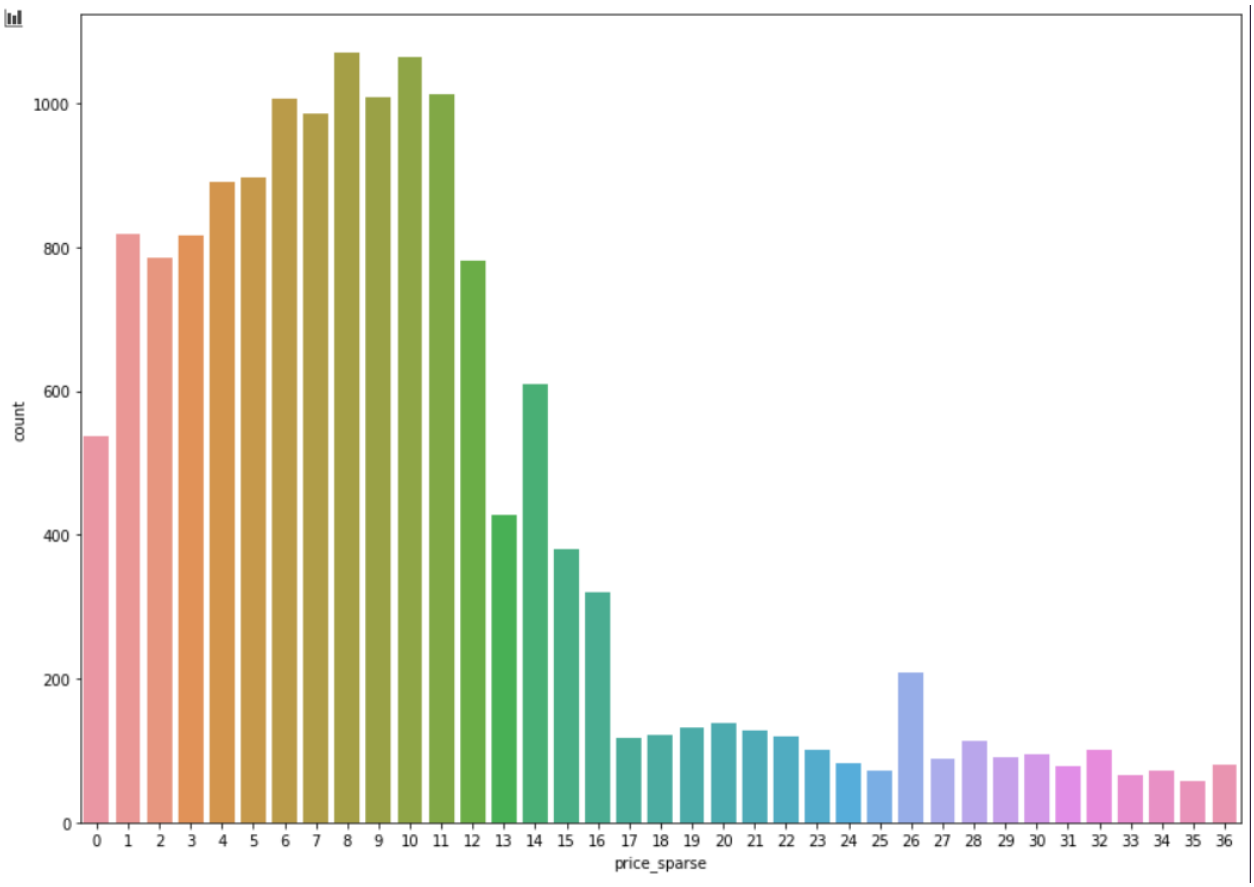
Meaning we're directly predicting the price with a linear (or relu) activation function

And as for the classification, there was this idea to categorize the prices into 37 distinct ranges, starting from the minimum range which was (195000, 245000) and going to the top prices with ranges of 50 000 dollars in each

Each of these 2 has been applied to CNN and ANN(s) individually as we'll see

---

After classifying the price we would have this distribution chart :



The first variation :

Both ANN and CNN are regression models :

This model is anticipated to use 2 distinct

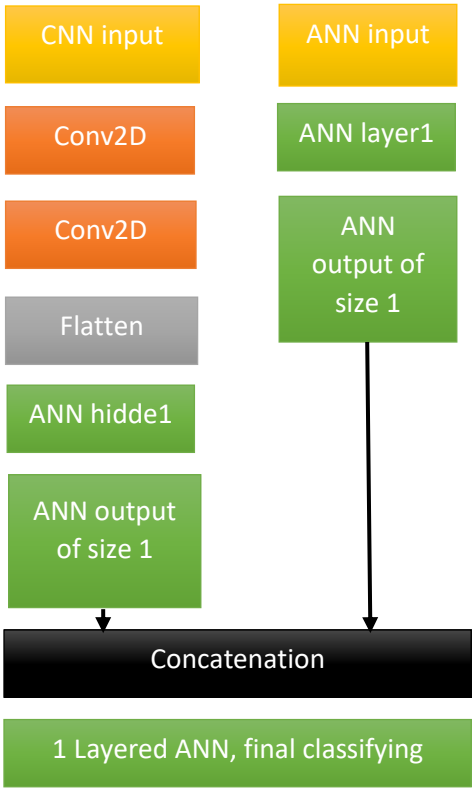
Regression models (CNN and ANN) to predict

A price individually and a final ANN decides

Which prediction between the CNN and ANN

Would have to be more emphasized as this

ANN(final layer) only have 2 weights

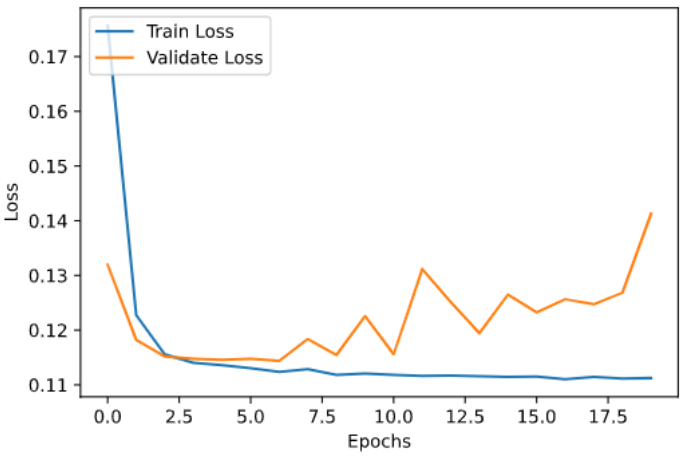


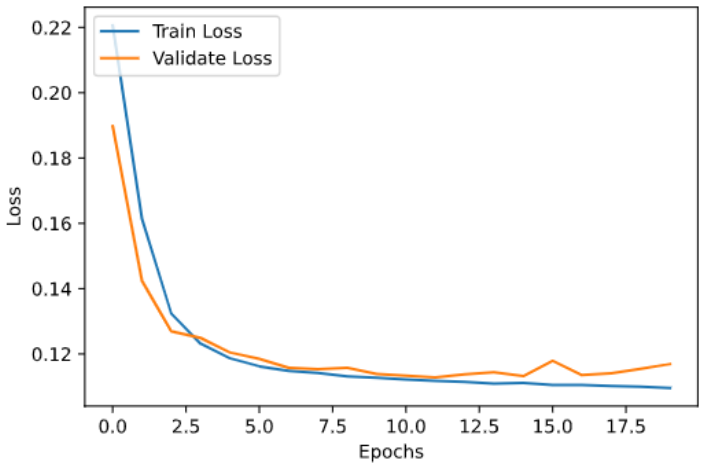
We were getting overfit (the right one is

the one with lower learning rate(0.0001 < 0.001)

which seemingly solved the issue

The results are :





The problem with this network was that we wouldn't have any predictions which looked over both the pictures and the info, we would want something that is a decision made upon the merged data

Varation 2 :

This one is exactly like the previous version

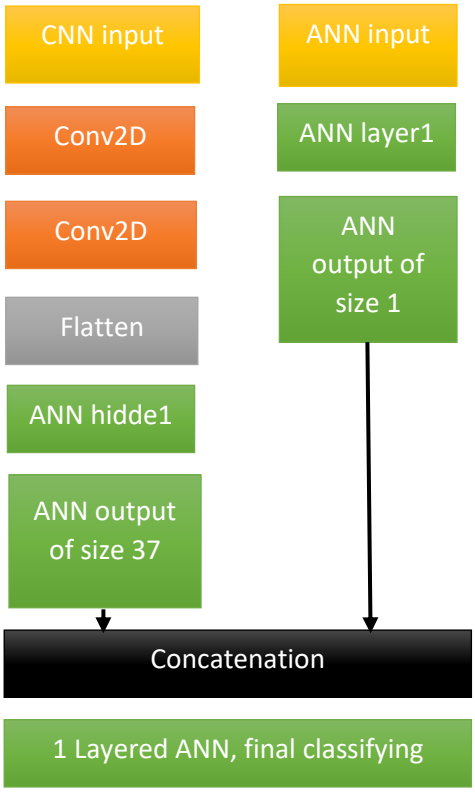
Only this time we have a classification model

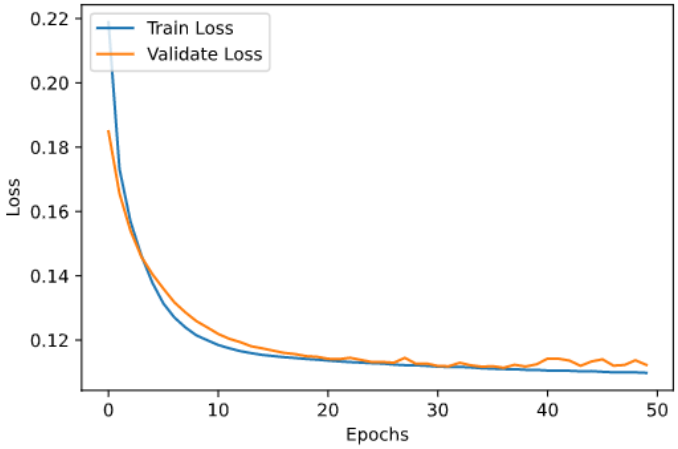
On our CNN output, and a regression model on

Our ANN output, honestly I don't expect this

Variation to be better than the last but

Just for the sake of learning I tried it ;\



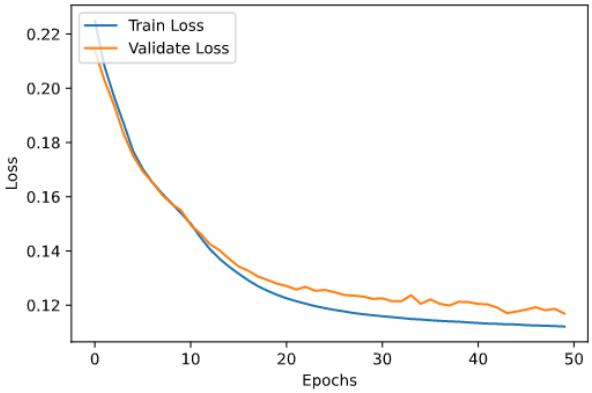


It actually had a decent convergence rate and didn't overfit as much as the last one so, we'll call it a worthy try :)

Variation 3 :

This variation Is when we'll have both the models as classification models which

The chart is exactly like the previous one so I'll just put the loss rate here



As you saw there were not any actual difference and improvement in the last 3 variations, so we'll try to come up with something new

variation 4 :

I know it seems a bit silly, but it actually

Helped a little (around 20K less

Error on mean of prediction errors)

Maybe the reason it had a little

Improvement was that the ANN

Structure got more complex comparing

To the First ANN but in this structure

We wouldn't have overfitting comparing

To the version that we just make the

First ANN more complex thus reaching

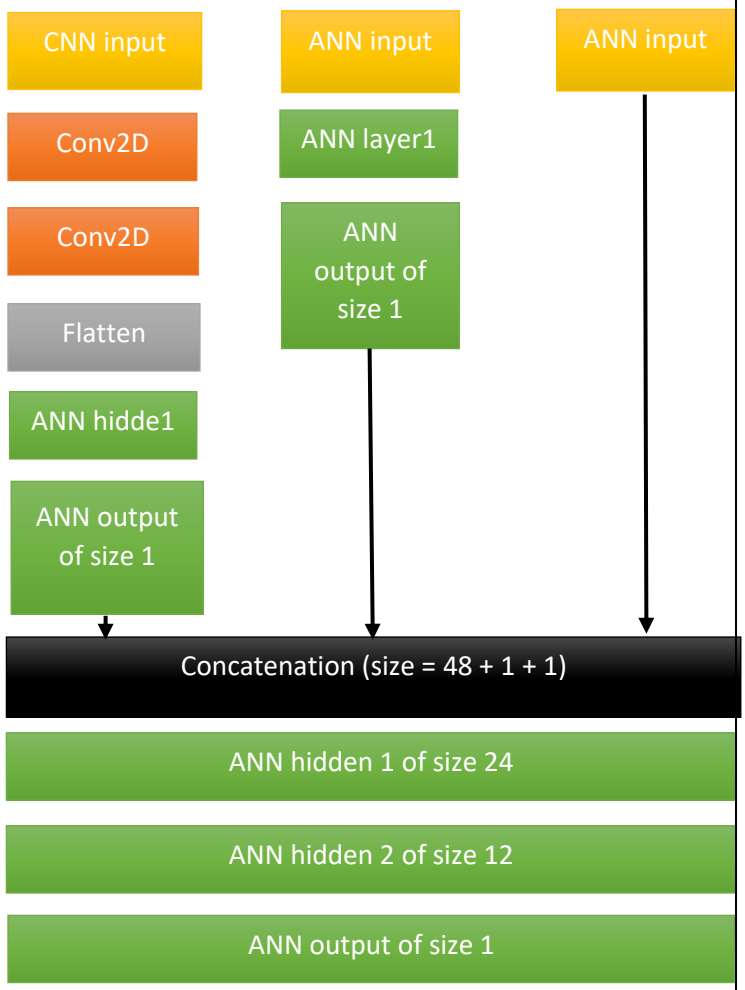
Overfitting

(and by the way I made the Conv2D

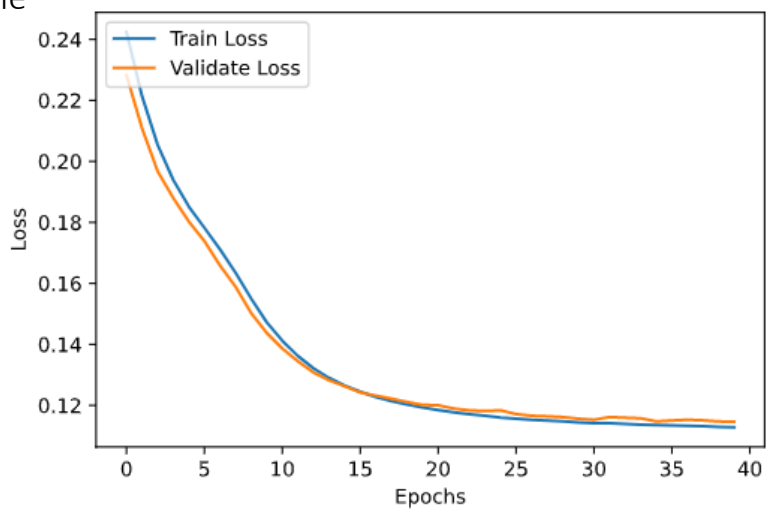
Layer from 2 layers to 3 layers, and

Tried both, the 3 layered one had a little

Better results (around 5K less mean error)



The error rate :



And finally we'll show the pictures of 9 random houses, with the Real and Error Price, in predicted data set

Real Price : 492
Error : 92

Real Price : 336
Error : 12

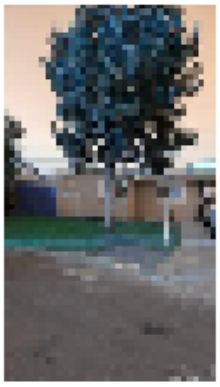
Real Price : 542
Error : 60

Real Price : 337
Error : -40

Real Price : 392
Error : 28

Real Price : 780
Error : -311

Real Price : 171
Error : -49



Real Price : 282
Error : 21

Real Price : 149
Error : 162