**1-LiDAR Image and Education**

Light Detection and Ranging (LiDAR) is a distance technology that shoots out light to the ground and detects the time required to reflect the pulse back to sensors. This image can be used to indicate the local geographical map.

**Question and presumption:**

Our topic is to investigate if we could relate the educational attainment to the LiDAR image's details, that is, the relationship between the local terrain and the local education deprivation.

The presumption is, they might be highly correlated when a larger area is covered by the LiDAR image, for example, municipal level. Otherwise, there might be not sufficient information when the area covered is only a community size or smaller. Since our images only contain a small area, some of the images can be highly predictive and some of them are not.

**Explanation:**

If the area covered is small, there might not be enough information to learn. Assume we have a LiDAR image containing a part of a factory and another image containing a part of a school. The information the picture conatins is very limited, such as types of buildings, and area of empty space, etc. A factory can look very similar to a school in a LiDAR image; however, the education deprivation of these two areas is absolutely different.

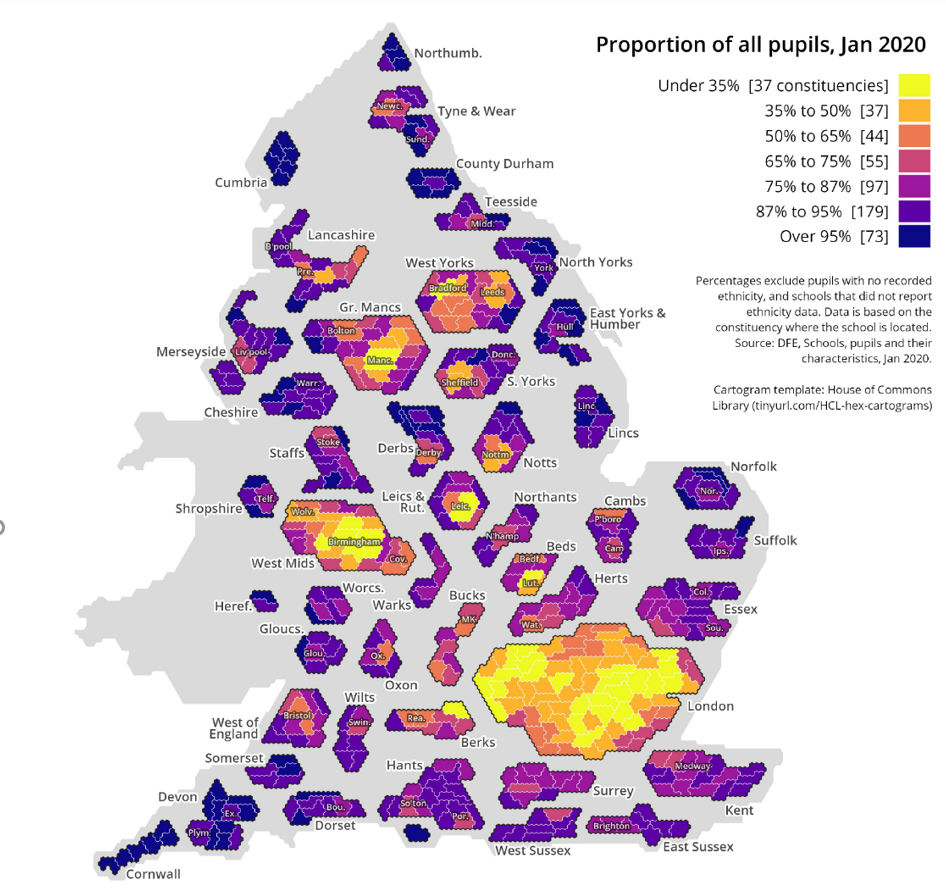
We clarified that some of that is still very possible to be predicted because certain types of local terrain may be highly correlated to education attainments. The school zone house is one example. According to the report published on redfin **(Unger 1)**, local property prices are largely affected by the performance of local schools. Below is a graph that concluded the situation in America.

图表, 条形图

描述已自动生成

We noticed that the house price is generally higher for the higher-ranked schools. Hence, we can conclude that some locations do have a relationship with educational attainment. LiDAR images indeed will help us to identify those locations based on terrain similarities.

Lastly, images covering a larger area could better help us determine educational attainment. A report published by the UK parliament divided the whole of Britain into several areas (parliament.uk 2021).



From the plot above, with this large scale of areas, the school pupils density (can be related to the educational attainment) becomes quite obvious. It is easy to predict the education index if we had a more macro image. Some of the cities have undoubtedly higher densities than others. If our LiDAR images contained this level of information, the model could precisely predict which part of the observation is in a particular area. In this way, the education index can be better indicated by the LiDAR images.

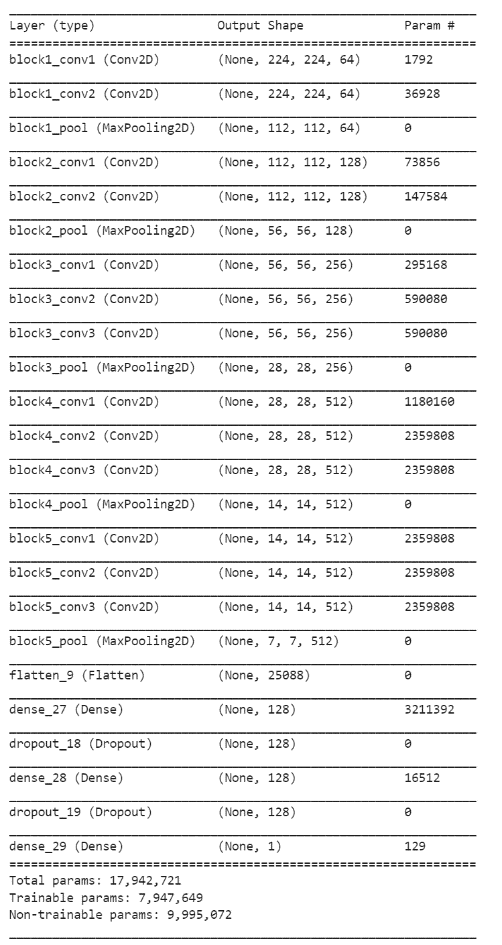
**2-CNN Modelling**

Two CNN models were taken into account here, one is VGG16, which is from the VGG family, and the other one is the Resnet50V2 from the Resnet family.

A new column called 'Path' was added to the embedded dataset, in this way we can connect our LiDAR images to the observations. Training and test datasets are created by a split of 0.7, and a seed of the Student ID was set.

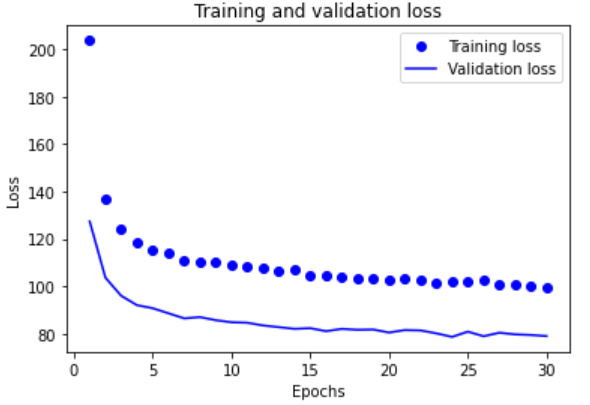
**VGG16**:

The VGG16 has 16 layers which included 5 blocks of convolutional layers. And we did not include the original dense layer, instead, our own designed three fully-connected layers were added to the model. Different parameters and dropout in the classifier were tried, such as 64, 128, 256 for neurons and 0.4, 0.5, 0.6 for dropout. It turns out that the combination of 128 and 0.6 for neurons and dropout respectively has the best solution. Since the target is regression, we will only have one neuron for the last dense layer with the activation function of "linear" in the model. We set our self-designed dense layers, and the last two layers before the classifier to trainable. Now we have the following architecture.

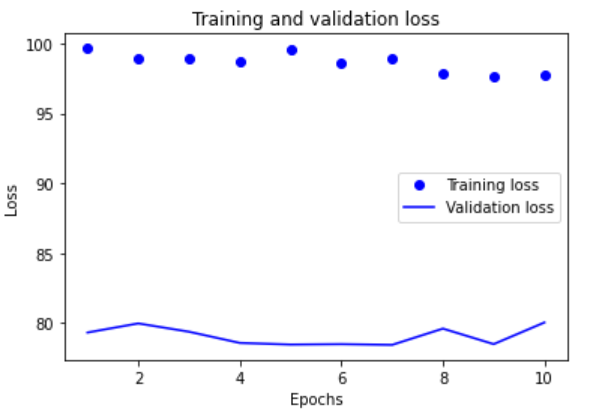


For the data generator, we first need to rescale the input to 1/255, as VGG cannot deal with unnormalized data. 20% of the training is kept as validation set. For the purpose of randomness, we set the zoom range to 20% because we might want to be more specific about the picture. And we allowed images to be flipped both horizontally and vertically because sometimes we need to investigate the situation when the building is located at another spot. We remained unchanged in terms of shearing since it would not make much sense to shear the image if all of them were plane plans that use colours to indicate three-dimension.

We used Adam with a learning rate of 0.000001 and a decay of 0.0001/200 as the optimizer. Adam is very efficient and straightforward to implement. This was tested multiple times and 0.001/200 is the number we found that works the best. We also need callback functions, which can greatly help the process of training. Earlystopping and Modelcheckpoints were selected, such that we will always be able to stop training when needed, and restore the weight. We used mean\_squared\_error as the loss function because we might expect negative indexes as a result (Even though we knew all the education indexes are all positive in our dataset, it could be zero in real life). The steps per epoch are determined by the sample size divided by our batch size of 64(determined by the Graphic memory). It turns out that, 322 steps for the training and 81 for the validation.



We tried 30 epochs first. Both the validation loss and training loss were constantly dropping. It seems like it has not yet come to convergence, so another ten epochs were added.

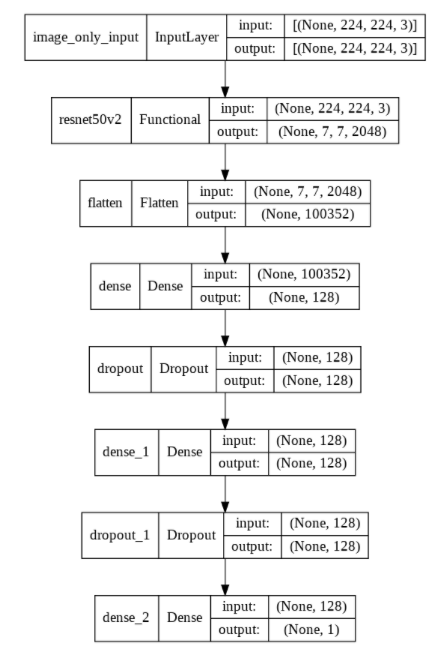


Notice that at the end, the validation loss became inconsistent and had an increasing trend; thus, it may indicate overfitting. Training should be stopped, and we hit our best solution at the 35th epoch with an MSE of 78.45. Loading this weight to our model would be a good idea.

Finally, we now input the test set to our trained model, and the prediction has an MSE of 82.89 compared to the true value.

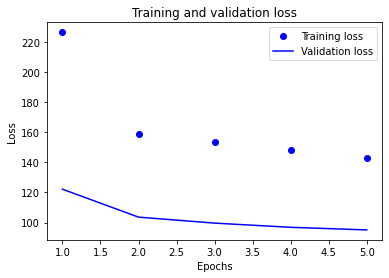
**ResNet50V2**:

ResNet50v2 is a model with more complexity. It is a modified ResNet50 and performs better on ImageNet. We have done almost the same thing here as we building the VGG16. We did not include the original classifier and added our own dense layers with the same parameters as the VGG (for similar reasons). And we used Model API to add the layer.

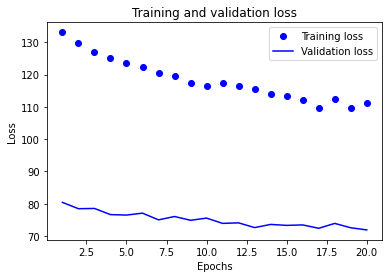


The same optimizer but with the decay of 0.0001/100 (tested multiple times and this returned best convergence) and loss function was also applied. This time, we do not rescale the data for the generator, instead, we will use its own preprocessor for batch normalization. As always, 20% of the training set will be used for the validation test.

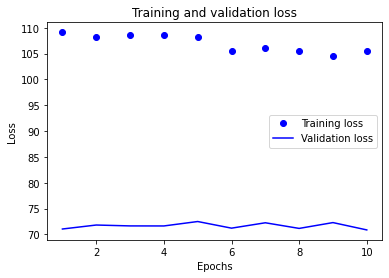
We now set only the dense layer as trainable but not the convolutional layers. We will warm up our model with just five epochs with only training the dense layer because the model will have a better performance in this way.



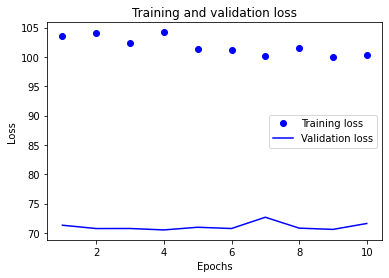
After the warm-up, we successfully reduce the validation loss to 98 from 120. Then we can begin our actual training. By setting all the convolutional layers to trainable, we should be able to further reduce the validation loss. We will do 20 epochs first.



The validation loss returned does not look extremely optimistic, but a decreasing trend is observable. We will now do 10 more epochs to try to converge.



The result this time looks fine but just a bit fluctuating. There is a necessity to try another 10 epochs.



The training loss is slowly decreasing, and the validation loss seems to converge at 70 when epoch 4.

Among those epochs, we considered epoch 34 with 70.51 validation loss as our best weight. After loading the weight, a test error of MSE of 72.23 was returned on the test set.

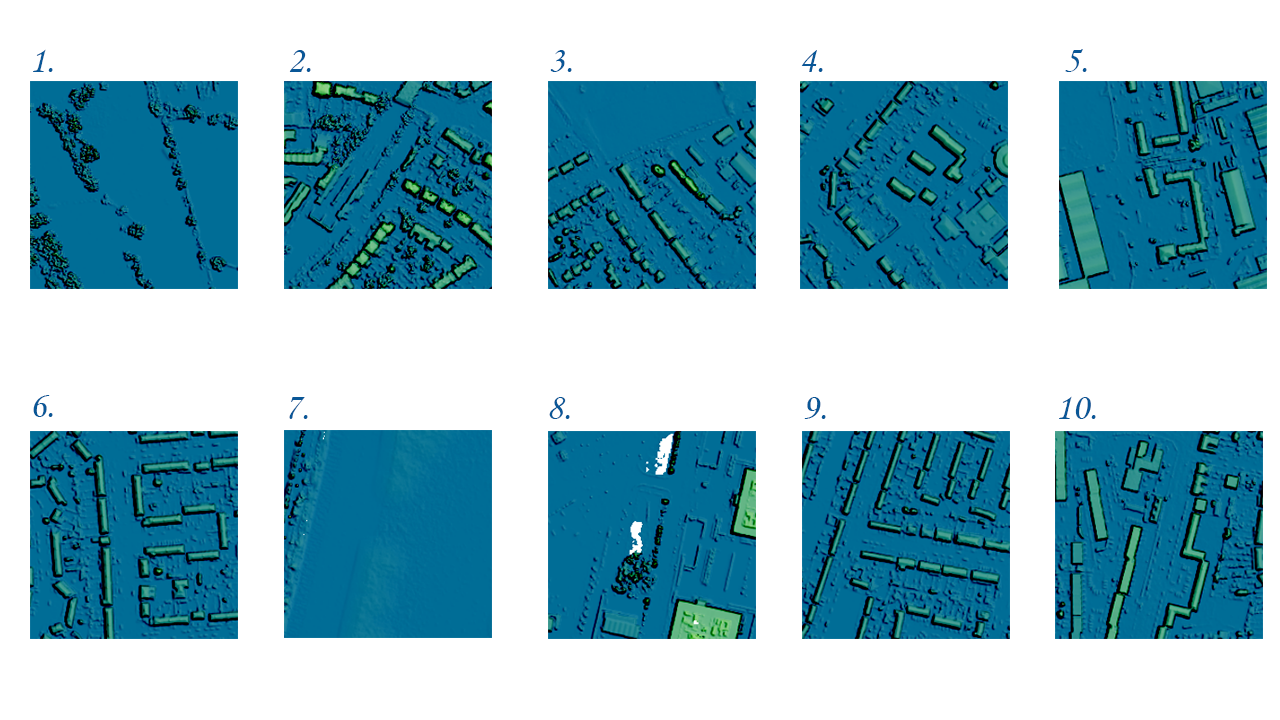
**Conclusion**:

For the test error, the ResNet50v2 outperforms VGG16, while VGG16 has 83 and the ResNet50v2 only has 72.23. In the VGG training, we noticed a validation loss convergence at 79 and the ResNet model fluctuates a lot more than VGG despite convergence. To address the issue, many different values of learning rate were also attempted, so do the parameters in both the training generator and the classifier. At last, the issue was still not resolved. Even though if we reduced the learning rate, the curve will become smoother, it would not come to a convergence state, and with much worse performance, so we decided to keep the learning rate as the original. There are two reasons for this situation. First, the ResNet50v2 is a more complex model, and we had all layers set to trainable. We probably did not try enough epochs to train. But because of the time, this may be the best we can do. Second, we have specified this idea in the previous section; the LiDAR image might just not be a strong predictive data for the education index. In conclusion, despite the fluctuation issue, our ResNet model still outperforms the VGG model. The ResNet is able to go more in-depth into the data because of its complexity.

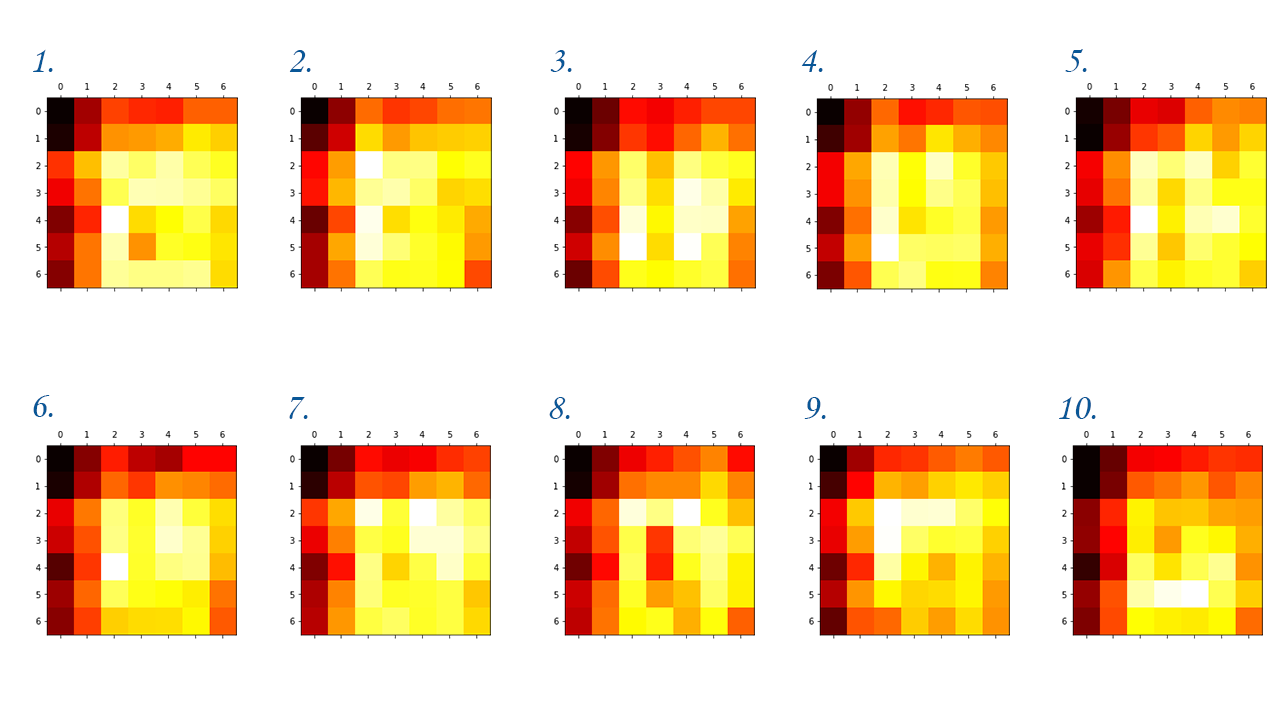
**3-GradCAM for heatmaps**

ResNet50v2 is chosen for this session since it performs better. We will create heatmaps for this model to investigate how it learns from images. To get the heatmap, we first need to recreate a new model with just the last convolutional layer of our original model, and this model will map the activation of that layer to the predictions. Next, we will calculate the gradient for the prediction of our selected images. Then we can take an average over all of the gradients for all maps. We will multiply each channel in the map array by its importance; then we have our heatmaps and need to normalize them between 0 and 1.

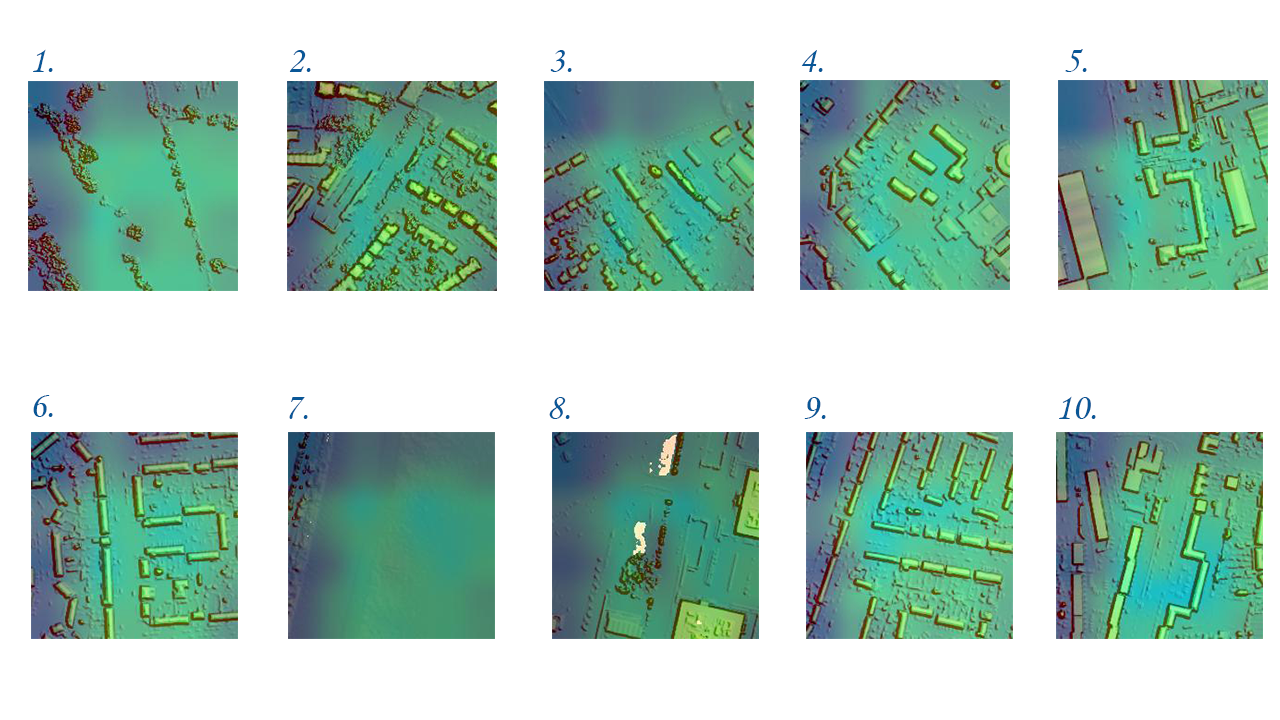
To cover the whole range of the education index, we divided our test observations into 10 levels based on its education index and selected the one with the lowest prediction error from each level to clearly see how it works. Below are the images selected.



After specifying the classifier and the last convolutional layer names, we noticed that all the gradients of pictures are negative, so we will have all zero values for the heatmap after normalization. To avoid this, we can take an absolute value of all the heatmaps since we only care about the gradients’ magnitudes. Then we need to reverse the colour otherwise the heatmap is inversed. We have the following heatmaps.



We can project them onto the original image so that we can better investigate them.



From our outputs, we noticed for images with a lower education index (1-6), the test error they have is very low. For the high education index, our model performs badly. Our model seems to only focus on the middle to the right side of the image. It also predicts all the images with similar patterns, and this might be the reason why it cannot predict higher index precisely. This second proved our statements earlier. Since the ResNet has a higher complexity and we set all the layers to trainable, it requires a far larger dataset and more epochs to be well trained.

**4-Conclusion**

Overall, using the LiDAR image to predict multidimensional deprivation seems to be a new approach. Indeed, after a decent amount of searching, there's not a lot of reports that have dabbled in this area. In our experiment, the LiDAR image is proven to have a specific ability to indicate the index; however, it depends on the type of the index. For example, the income index might be highly correlated to the details of LiDAR. Imagining the CNN learns to distinguish a community full of fancy houses from the LiDAR image, there is no way that a family with very low income living in there. In our case, the LiDAR image is very good at predicting areas with low education deprivation index; however, when it comes to a place supposed to have a large index, our model suddenly performs worse. This proved our presumptions, for education, LiDAR is good at predicting some parts, and it may look terrible for the rest. In general, the usefulness of LiDAR images in the data prediction area really depends on what our target is. LiDAR images also have advantages. The cost is low, and the technology nowadays is very mature to efficiently create LiDAR images.

Some governments have made the LiDAR image public. However, it might raise concerns if it were massively utilized **(Partika 2014)**. Privacy can be one of the biggest concerns that come with LiDAR images. For instance, most people may not want to be judged just by where they live, especially when everyone can access their data freely. There are two recommendations to resolve the issue. First, the government can form new departments, such that if any organization needs to access the data, they need to submit applications to the department and undergo an examination process. Second, governments can make the data have an expensive deposit threshold. If there were anyone that needed to access the data, they would need to put a large amount of deposit at least to reach the data. This can prevent small unauthorized organizations to access the data freely.

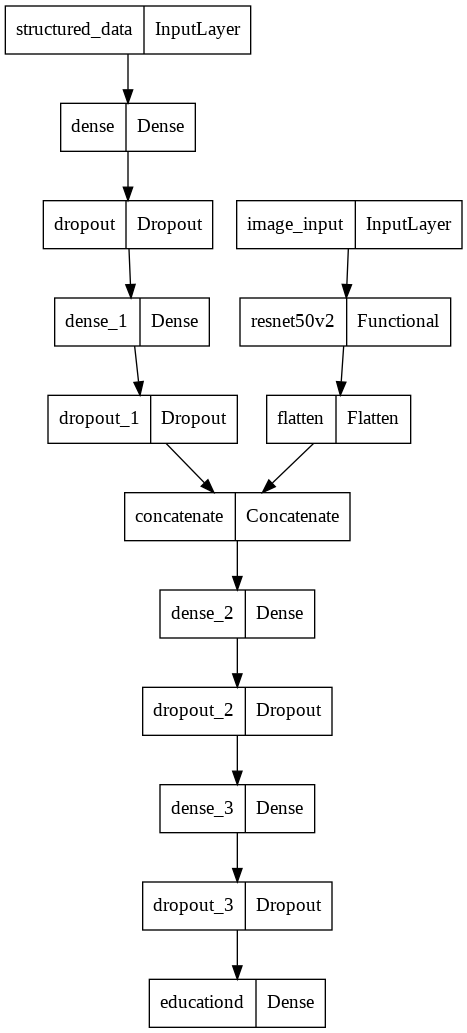
**EX-Multimodal**

**Data preprocessing:**

For this multimodal, we will treat our data differently. We will take columns more than just education into account. We noticed that some of the categorical features we have contained the same observation as others, but just in the string form. For this reason, we dropped columns 'LSOA11NM', 'SOAC11NM', 'MSOA11NM', 'LAD17NM', 'LACNM'. We noticed that LSOA11CD, SOAC11CD, MSOA11CD, LAD17CD had too many categories and we are not able to handle all of them. A special treatment was applied to them in the train set. Our target is education, so I ranked the categories for each of the features by the average education index within each category. Then I re-categorize those four features into fewer categories based on that, for example, categories with top five averaged education deprivation will all be labeled as 1. The same treatment was also applied to the test, but the rank of categories is based on the training set, such that we won't have an actual result leak. Then we used get\_dummies to make all categorical variables as dummy variables. The next step is like what we have done before, for both test and training set, we added a column called 'path' that contains the correct image path for each observation. Indeed, we also need to minmaxscale our continuous columns, which includes "income", "employment", "health", "crime", "barriers", "living\_environment", for both training and test set.

**Modelling**:

The architecture for this multimodal is kind of different. A picture of that is shown below.

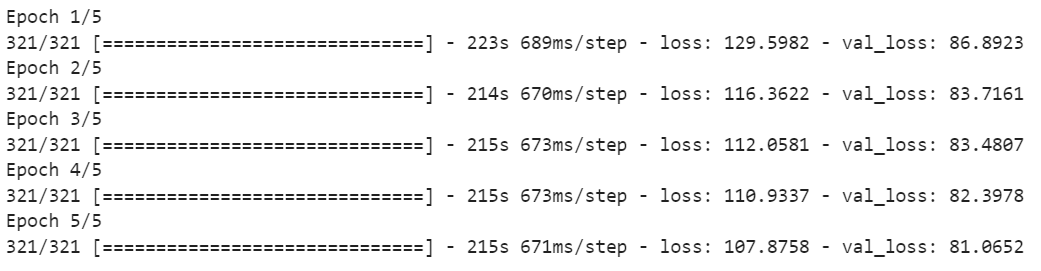


We now have two legs for each of the embedding and LiDAR images. In our code, we have two dense layers for the structured data with 64 and 32 neurons respectively because of the dataset size we have, and both of them have relu as the activation. There are also two dropout layers with a size of 60%. For the unstructured data, we will use the same model as what we have done before, the ResNet50v2.

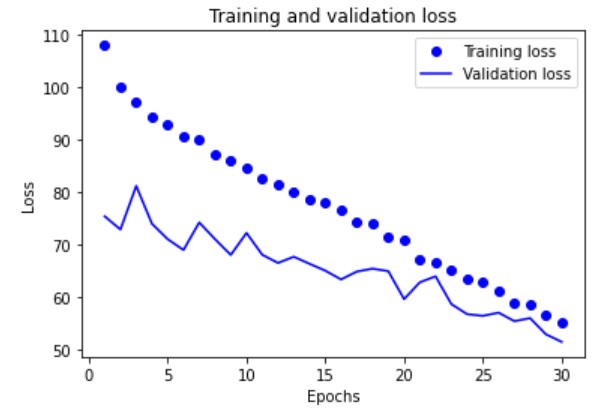
Next, we concatenate those two legs together to have our top. Just like before, we have three dense layers, the bottom two have 128 neurons, while the top one only has one neuron because we need to output a regression. All of them have a dropout size of 60%, and they all used relu as activation. We only set the classifier as trainable.

We are going to use the same optimizer as before, Adam with a 0.000001 learning rate and decay rate of 0.0001/200. When it comes to the data generator, we will set the y column as all the columns except the image path. We also specified our target location as 2 (the column of education), and constantly gave images each time for all train, validation, and test generator. Just as before, we have the steps per epoch of 322, calculating from sample size divided by the batch size we have, and validation steps of 81. The metric we used is still the mean squared error. Reasons have been previously explained.

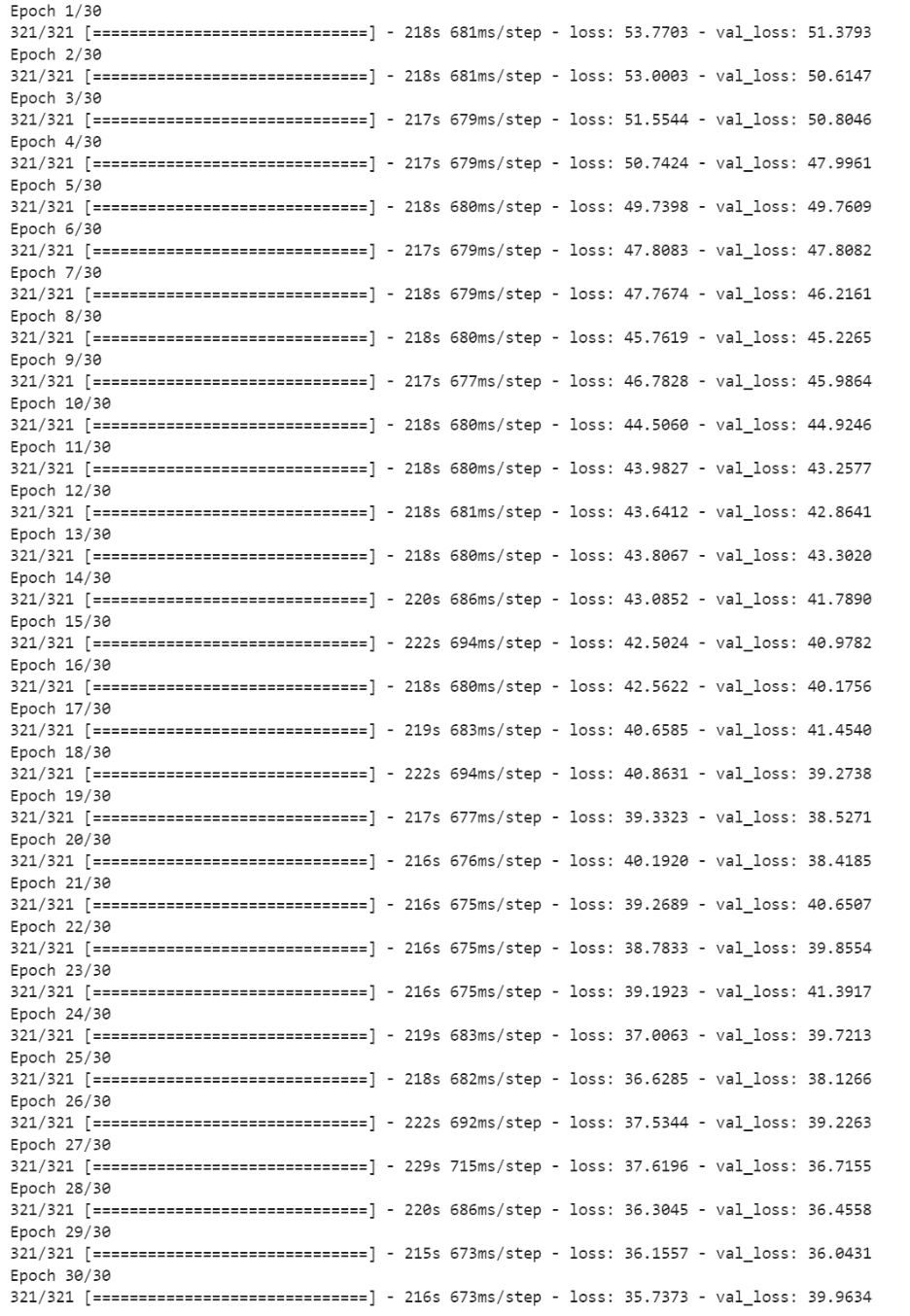
Again, we believe a 5-epoch warm-up will greatly help the process of training.



The validation loss dropped after the warm-up. We then set all the convolutional layers as trainable, and start the real training process of 30 epochs with help from callbacks including Earlystopping and Modelcheckpoint. The patience for earlystopping is 5, since after 5 epochs of increasing, we will know the model is overfitting.



We noticed that the validation loss is constantly dropping, however, it is quite obvious that it is far from reaching its convergence. Another 30 epochs of training will be needed.



Now we noticed that our model comes to convergence at around validation loss of 38. We will load the weight from the 59th epoch since it has a minimum loss of 36.04. The test step size will be the test sample size divided by our batch size, and with this, we can input our test set to the trained multimodal model. We calculated the mean square error of our predictions, which is 37.96. This multimodal model outperforms the VGG and ResNet so much with the help of structured data.



Word count (excluding the multimodal): 2200

References:

*A complete guide to LIDAR: Light detection and ranging*. GIS Geography. (2021, October 29). Retrieved December 15, 2021, from <https://gisgeography.com/lidar-light-detection-> and-ranging/

ParliamentUK. (2021, June 22). *The influence of Place*. The forgotten: how White working- class pupils have been let down, and how to change it - Education Committee - House of Commons. Retrieved December 15, 2021, from <https://publications.parliament.uk/> pa/cm5802/cmselect/cmeduc/85/8506.htm

Partyka, J. (2015, February 28). *Why data from Automated Vehicles Needs Serious Protection*. GPS World. Retrieved December 15, 2021, from <https://www.gpsworld.com/why-data-> from-automated-vehicles-needs-serious-protection/

Unger, T. (2020, October 6). *Affording a house in a highly ranked school zone? it's elementary*. Redfin Real Estate News. Retrieved December 15, 2021, from https://www.redfin.com /news/paying-more-for-a-house-with-a-top-public-school-its-elementary/