**Interpolation for Medical Imaging using a Multi-Plane PixelCNN**

**Abstract**

**Introduction**

One of the major challenges in using medical images to do predictive analytics is the variation in the data, including the data quality. MR and CT imaging rely on creating images in slices taken of the axial plane, from the head down. These can then be stacked to create the coronal view, the view from the front, or the sagittal, the view from the side. Variation in medical imaging can be in slice thickness, which can usually range from around 1mm to 5mm in slice thickness, and the spacing between the slices. These types of variations can have a dramatic affect on the capabilities of predictive of models. One method to overcome some of the issues is to use interpolation in between the slices of higher slice thickness images to increase the resolution of medical images a long coronal and saggital planes. Commonly used interpolation methods include the Nearest Neighbors, Bilinear, Bicubic, or Windowed Sinc methods.

The problem is unique to the standard interpolation problem because the images are in fact three dimensional in nature and require slices in between to be interpolated, whereas two dimensional images are typically dilated and the pixels in between are filled in using common algorithms. Typical interpolation algorithms function by guessing intermediate pixel intensities whereas slices in medical imaging the pixel intensities should be consistent with the type of object that is in the image, but in a different location or position.

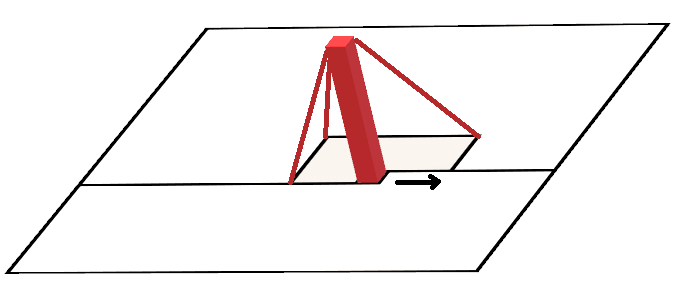
Another problem is being able to capture the underlying texture of the images which is very important for the use of radiomics being able to make predictive models. This can be difficult to achieve using certain types of generative models since models such as Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE) do not provide a means to calculate likelihood, which makes it difficult to explicitly take into account textures during training.

**Background**

A PixelCNN is one of the three major types of generative models used in Deep Learning which also includes VAE and GAN achitectures. PixelCNN differs quite from the other generative models. PixelCNN uses an explicit density function that use an explicit specification of the distribution of the random variable [1], and most models in machine learning and statistics are in this form [2]. GAN models use an implicit density function in which a generator implicitly defines a probability distribution based on a latent vector [3]. A PixelCNN can also be distinguished from other models because it uses a tractable density function that optimize the likelihood of the training data, whereas a VAE defines an intractable density because it makes either a variational approximations, Monte Carlo approximations, or both [4].

PixelCNNs also differ greatly from other generative models because the optimize the likelihood of training based on the individual pixels [5]. PixelCNN is an autoregressive model and predicts pixels one by one and bases the prediction based on previous predictions [5]. Its function can be expressed as the product of the probabilities of a pixel based on all previous pixels [5]:

A PixelCNN makes a prediction based on the intensity of the pixel, see *Figure 1*. The original method for doing this was to take softmax for a range of 256 pixel intensities which requires and output of 256 channels. This becomes quite resource intensive



A problem discovered early during the evolution of the PixelCNN is that there is a blind spot in the receptive. This is a result of using a mask in a convolutional layers kernel.

**Method**

The proposed model is a slightly modified PixelCNN. This model receives multiple slices as inputs into the model. Training is done using three slices of imaging data at a time using source imaging data with half the slice thickness of the intended images to be up sampled. During the data preprocessing phase, images are divided into three slices, which includes the top slice, the bottom slice, and the target slice.

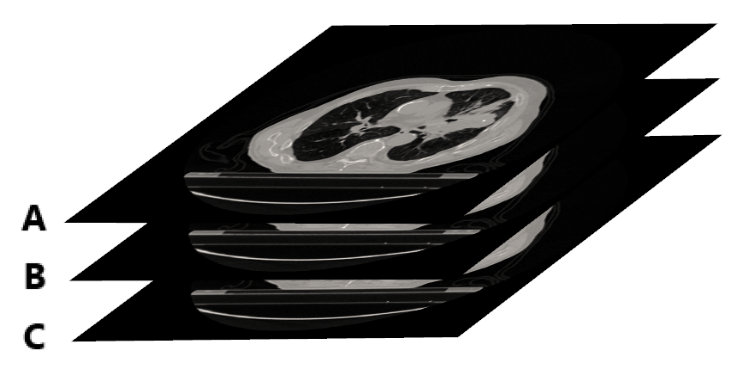


Figure : Where A is the top slice, C is the bottom slice and B is the target slice.

During training the loss is calculated by every pixel of all three planes, but at generation, only the predicted slice is used to make the prediction in the target slice. Furthermore, it is important to rearrange the slices so that the target layer is the last layer to be evaluated and thus the top and bottom layers will ultimately contribute the prediction of the target layer. Having the correct order not only improves the overall training but it can also prevent the problem of just copying the top slice in certain instances.

At generation, the model is run through a forward pass to make a prediction pixel by pixel. The generator receives only the top slice, and a bottom slice while creating an empty slice with no information. Image generation proceeds pixel by pixel with all three slices being fed into the generator, but only updating the target slice.

**Results**

**Conclusion**

**References**

**[1] Diggle and Gratton 1984, “Monte Carlo Methods of Inference for Implicit Statistical Models”**

<https://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1984.tb01290.x>

**[2] Shakir and Balaji 2017, “Learning in Implicit Generative Models”**

<https://arxiv.org/pdf/1610.03483.pdf>

**[3] Ian J. Goodfellow 2014, “Generative Adversarial Nets”**

**[4] Ian Goodfellow 2016, “Tutorial: Generative Adversarial Networks”**

**[5] van den Oord, Kalchbrenner and Kavukcuoglu 2013 “Pixel Recurrent Neural Networks”**