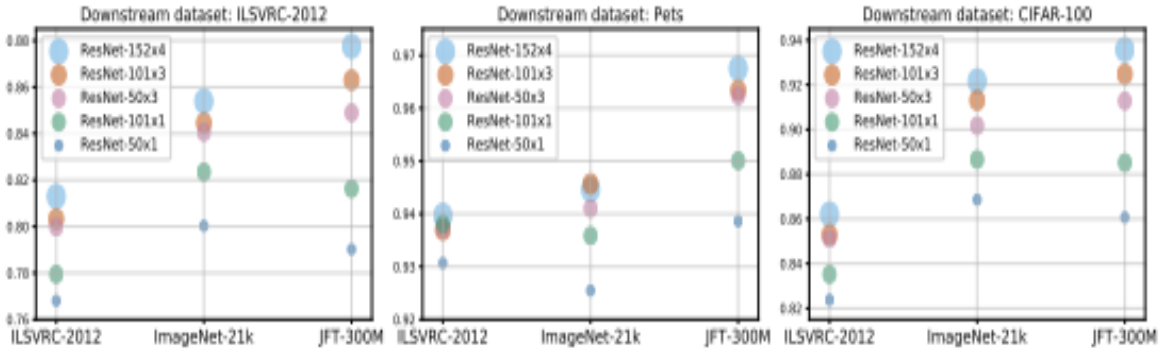
2. Write a 2 page (excluding pictures) summary of what architectural improvements, if any, the authors have assumed on top the ResNet-xyz V1 architecture and why (20 points)

Architectural improvements are divided into two parts:

a) Upstream Pre-Training
There are two components to pre-training. First one is scale. You need to have a lot of data and a lot of models.

The consensus is that larger neural networks result in better performance. We investigate the interplay between model capacity and upstream dataset size on downstream performance. We evaluate the BiT models of different sizes (ResNet-50x1, ResNet-50x3, ResNet-101x1, ResNet-101x3, and ResNet-152x4) trained on ILSVRC-2012, ImageNet-21k, and JFT-300M on various downstream benchmarks. These results are summarized in Figure below



X-axis: Datasets to pre train on ranging from small(ILSVRC-2012) medium(ImageNet-21k) to large dataset(JFT-300M)

Y-axis: Accuracy

ResNet-XYZ are the models. The larger the circles the larger the model's architecture.

Its noted that the larger the model's architecture, the better the accuracy in most cases. Also, the larger the amounts of data(x-axis) the better the performance.

The larger the model's architecture and the larger the datasets, the better the accuracy.

In some cases, its observed that having a small model architecture and a large dataset reduces the accuracy.

Belkin's double descent curve hypothesizes the relation of the number of parameters to dataset size **#P/#D** vs **Validation loss** (V.L). Adding more parameters to the model reduces the validation loss up until it starts to overfit at the **interpolation threshold (I.T)** which is equal to 1 **(#P/#D = 1**) number of parameters is equal to number of datapoints after which it comes down for several reasons.

Adding more data and or parameters shifts the point. Which might explain the behavior of the small models v. increased data.

Second component is Group Normalization (GN) and Weight Standardization (WS).

Batch Normalization (BN) is used a lot. BN involves having many datapoints, putting through your layers (CNN) to an intermediate interpretation, you calculate the mean and variance of each of the features and adjusting it to get to a mean of 0 and standard deviation of 1.

This is dependent on batch size, how many datapoints you have.

However, we find that BN is detrimental to Big Transfer for two reasons. First, when training large models with small per-device batches, BN performs poorly or incurs inter-device synchronization cost. Second, due to the requirement to update running statistics, BN is detrimental for transfer.

Instead we can break the batches into groups, and distribute them into many machines (Data parallelism) with many TPU's (Tensor Processing Units)

This paper uses a batch size of 4000 distributed to 500 TPU's giving 8 samples per batch, which is not very good for batch normalization.

A fix for this is group normalization (GN) and weight standardization (WS). They don't require the other samples of the batch. Works on a per sample basis and normalize the features within groups of each channel. Weight Standardization (WS) standardizes the weights to be of a normal distribution. They allow you to not have to synchronize constantly between workers at training time which makes everything run faster and not have a problem where you have 8 samples per worker.

b) Transfer to Downstream Tasks (Fine Tuning-FT)

We use a heuristic rule—which we call BiT-HyperRule—to select the most important hyperparameters for tuning as a simple function of the task's intrinsic image resolution and number of datapoints. We found it important to set the following hyperparameters per-task: training schedule length, resolution, and whether to use MixUp regularization.

BiT-HyperRule which is one hyper parameter, run through the rule, which tells you which the parameter should be which behaves akin to a look up table.

For fine tuning (FT), you only need to grid search over one hyper parameter, which then decides on training schedule length, resolution, and whether to use MixUp regularization.

MixUp regularization is a technique that helps when you have little data and it interpolates between datapoints and trains on datapoints from half one class and half another class to make more data available.

We found that MixUp is not useful for pre-training BiT, likely due to the abundance of data. However, it is sometimes useful for transfer. Interestingly, it is most useful for mid-sized datasets, and not for few-shot transfer.

Surprisingly, we do not use any of the following forms of regularization during downstream tuning: weight decay to zero, weight decay to initial parameters, or dropout.

Only use weight decay during pre-training.