BTP Project

AR / VR based Furniture Marketplace

By - B21SK01

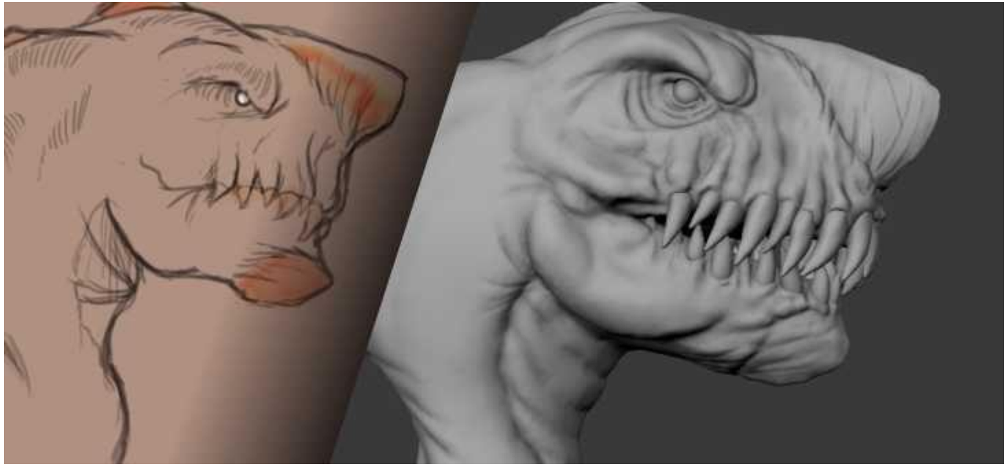
Sayam Kumar - S20180010158
Nitin Kumar Chauhan - S20180010119
Dasari Jayasree - S20180010047

Outline

- Objective
- Progress
- Live Demo
- Model details
- Mobile App details
- Major challenges we faced
- Results and Conclusion

Objective

- Use Augmented Reality / Virtual Reality to simplify the work of furniture showrooms and Interior designers.
- Scan furniture and store them in central place.
- Improve customer engagements by allowing them project furniture in rooms.



Progress

First Evaluation

- Learnt the basics of 2D-3D modelling.
- Literature Review.

Second Evaluation

Model creation phase

- Created a novel end-to-end pipeline from 2D image to 3D model shape.
- Represented the generated 3D model in virtual space.

Third Evaluation

Mobile App creation phase

- Created an end-to-end mobile app joining the outcomes of second evaluation.
- An image is taken from mobile and a 3D model is rendered in virtual space after being generated from server.

Progress

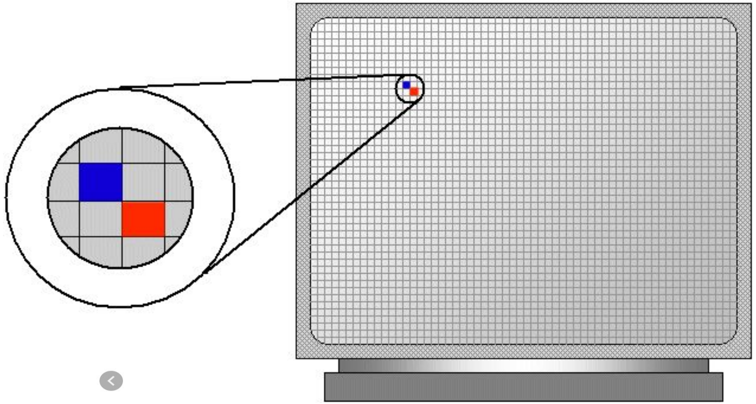
Fourth Evaluation
Fine tuning phase
• Trained the network on cabinets and sofa models. • Analysis of generative power of the network • QR code retrieval of models • Improved Mobile App to be fully optimized

Live Demo

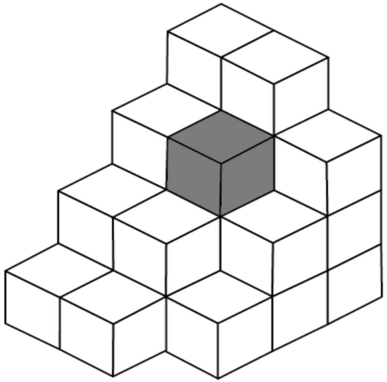
Model Related Work

Terminology

<u>Pixel</u> - Smallest unit for a 2D image. All images are formed by stacking and arranging pixels.



<u>Voxel</u> - Smallest cubical unit for a 3d structure.



Problem Definition

- Given an input Image (I), generate a 3D model (M) for it.
- This can be seen as a Supervised Learning problem. Let’s say, we make use of hypothesis function (h).
  - M = h(I)
- Further breaking it down,
  - We have ground truth Image I and it's true model M from the dataset.
  - Our network learns a transition from pixel representation of 2D image to voxel representation of 3D model.
  - We compare the generated 3D model with true 3D model and use sparse loss functions to train for visually appealing results.

Related Works

- Geometry-based Reconstruction - Make 3D objects from 2D images by satisfying geometry constraints and adding prior knowledge. Eg - If I am modelling sphere, I make sure the 3D model has equal radius from center to surface (Moll et al, 2015).

- Learning-based Reconstruction - These methods use data driven approaches for 2D-3D mapping. Initial works are described in 3D Recurrent Neural Network (3D-R2N2) (Choy et al, 2016) paper which uses multiple views of images and Long Short Term Memory (LSTM).

Our method uses data driven approaches with Encoder-Decoder networks as hypothesis function.

Dataset

- Used Shapenet dataset. Original shapenet:
  - Is built by generating 3D models using Computer Aided Design (CAD)
  - 270 categories of objects
  - 51k unique 3D models

- Our data pipeline -
  - 4 categories - chair, table, cabinet and sofa
  - Till 2^{nd} eval - 2.2k chair models, 2.5k table models
  - Now - Added 1.4k cabinets, 2k sofa models
  - 12 2D views for the 3D model as images captured by 30^{o} rotation around vertical axis.
  - Images in png format and 3D model in obj format.
  - Background from the images is removed, if any.

Learning Representation

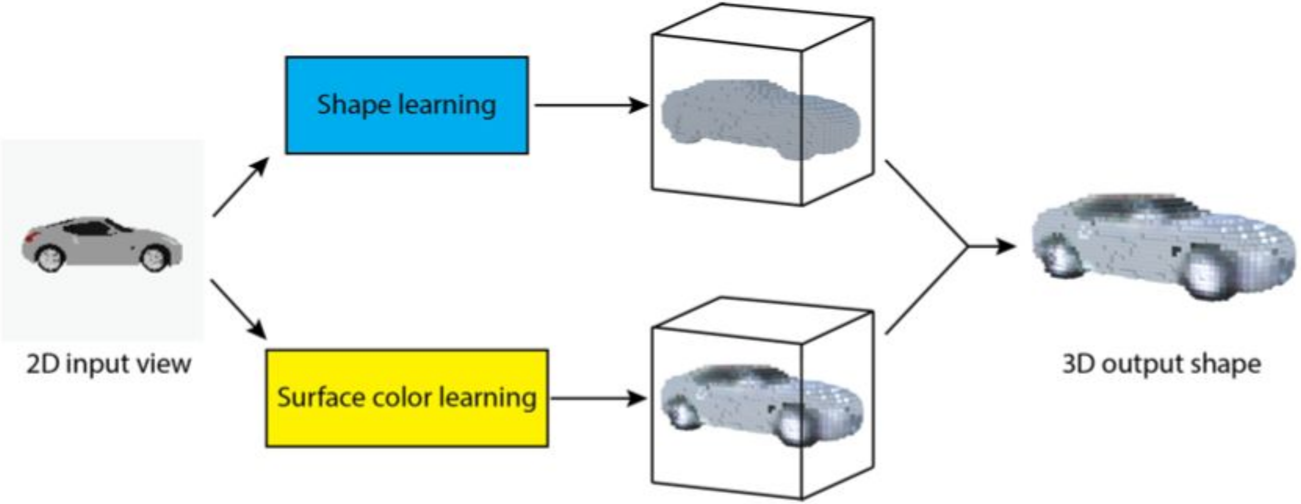
We need to learn two things from an image - 3D model shape and 3D model texture.

- Joint Learning
  - Learn both tasks simultaneously
  - Each occupied voxel is learnt by 0 or 1. 0 means unoccupied voxel and 1 means occupied voxel.
  - For color, the network learns three numbers for RGB for occupied voxels and a vector [-1, -1, -1] for unoccupied voxels.
  - Learn 4 numbers in total.
  - A bad approach - For sparse models (like chair) and imbalanced voxels, the network pushes more towards learning unoccupied voxels better.

Learning Representation

Our Approach

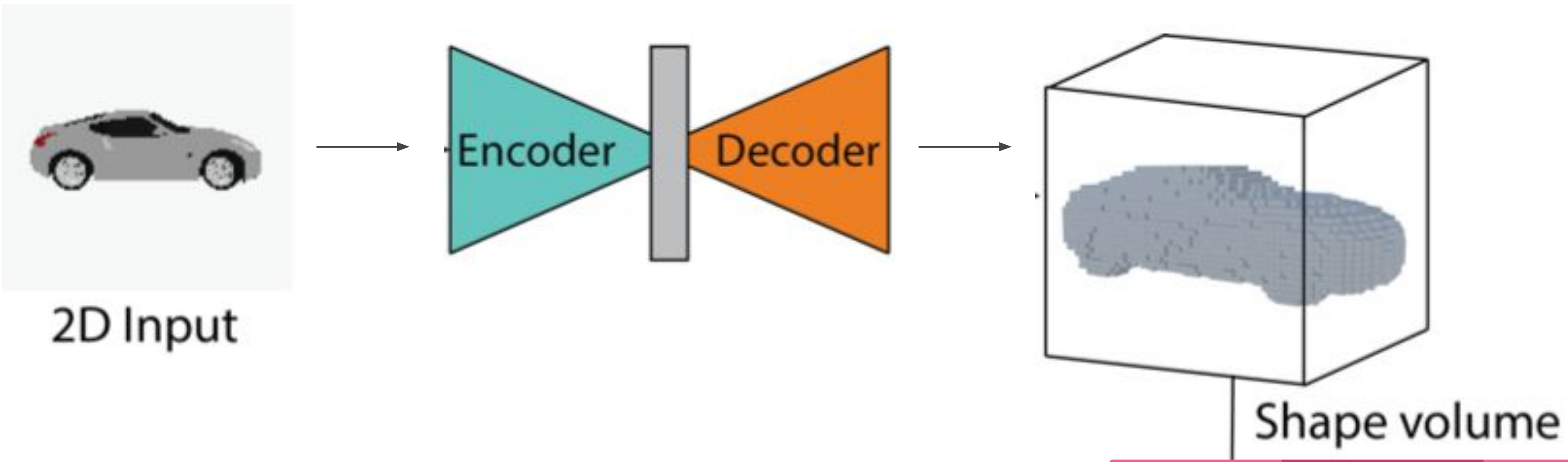
Learn both tasks separately.



This avoided the training difficulty due to shape color imbalance

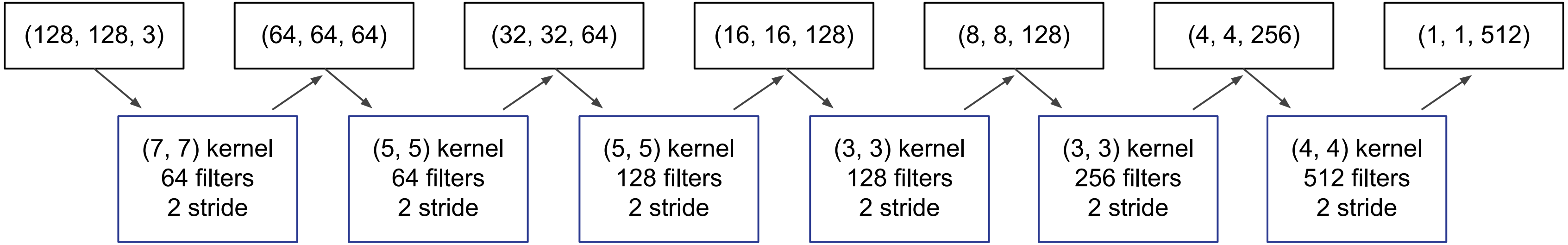
Shape Network

- Given an input image (I), the encoder network (e) first learns the latent representation and then is fed into a decoder network (d).
- The decoder generates a 1 channel shape volume V' = d(e(I))

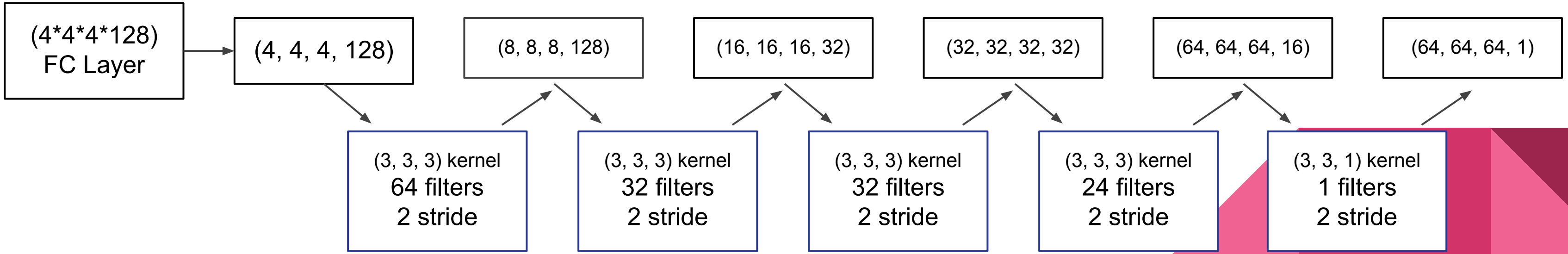


Shape Network

Encoder - 6 2D convolution layers (conv layers for weight sharing and focussing on local relations)



Decoder - 5 3D convolution layers



Shape Network Loss Functions

- L2 loss = $\sum_i (V_i - \hat{V}_i)^2$
  where i is voxel index and V_i is state of voxel occupancy {0, 1}
- Moved to Cross Entropy loss
  Loss = $-\sum_i [V_i \log(\hat{V}_i) + (1 - V_i) \log(1 - \hat{V}_i)]$
- To handle sparse relations better, we modified it to use two combined losses -
  - False Positive Cross entropy on unoccupied voxels
  - False Negative Cross entropy on occupied voxels

$$FPCE = -\frac{1}{N} \sum_{n=1}^{N} [V_n \log \hat{V}_n + (1 - V_n) \log(1 - \hat{V}_n)]$$

$$FNCE = -\frac{1}{P} \sum_{p=1}^{P} [V_p \log \hat{V}_p + (1 - V_p) \log(1 - \hat{V}_p)]$$

Training

- Train / Test Split = 0.9 / 0.1
- Adam learning optimizer
- Learning rate = 0.0003
- Batch size = 60
- Training Epochs = 500

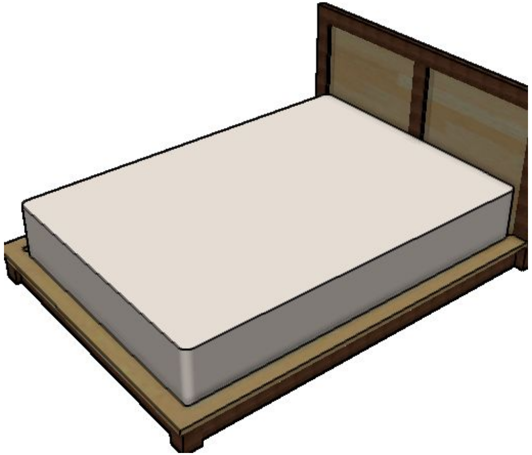
Tech Stack

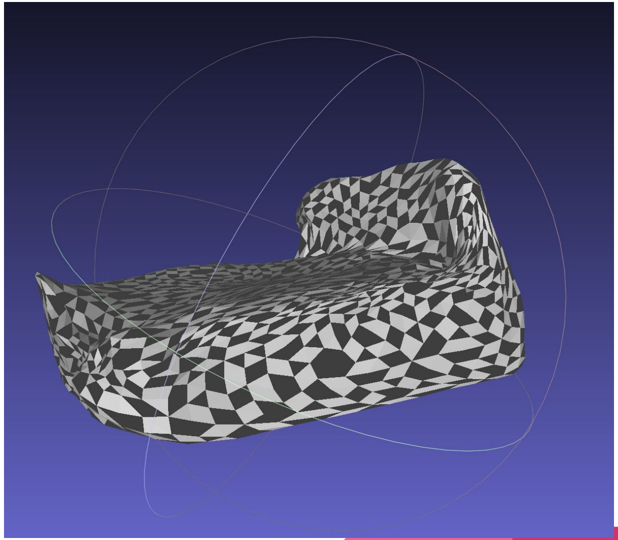
- Tensorflow
- cv2
- Scipy
- Python
- SSH keys protection

Performance Check on Out-of-Distribution Data

- The model is trained on 4 furniture categories i.e. chair, table, cabinet and sofa.
- We got visually appealing results for categories outside the 4 mentioned above.

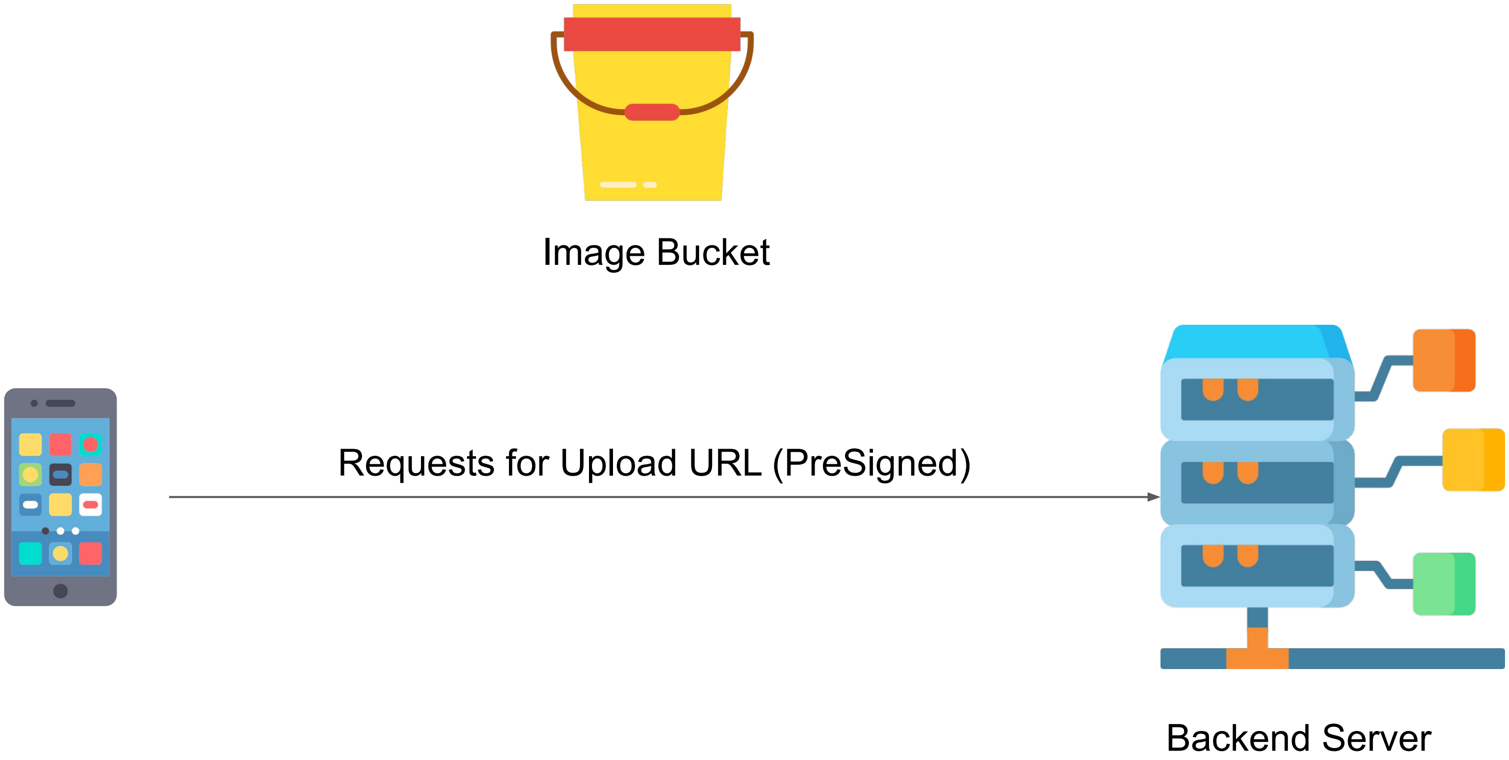
Examples shown in live demo and one more here: Checking model of bed





Mobile App Related Work

Mobile App Interactions



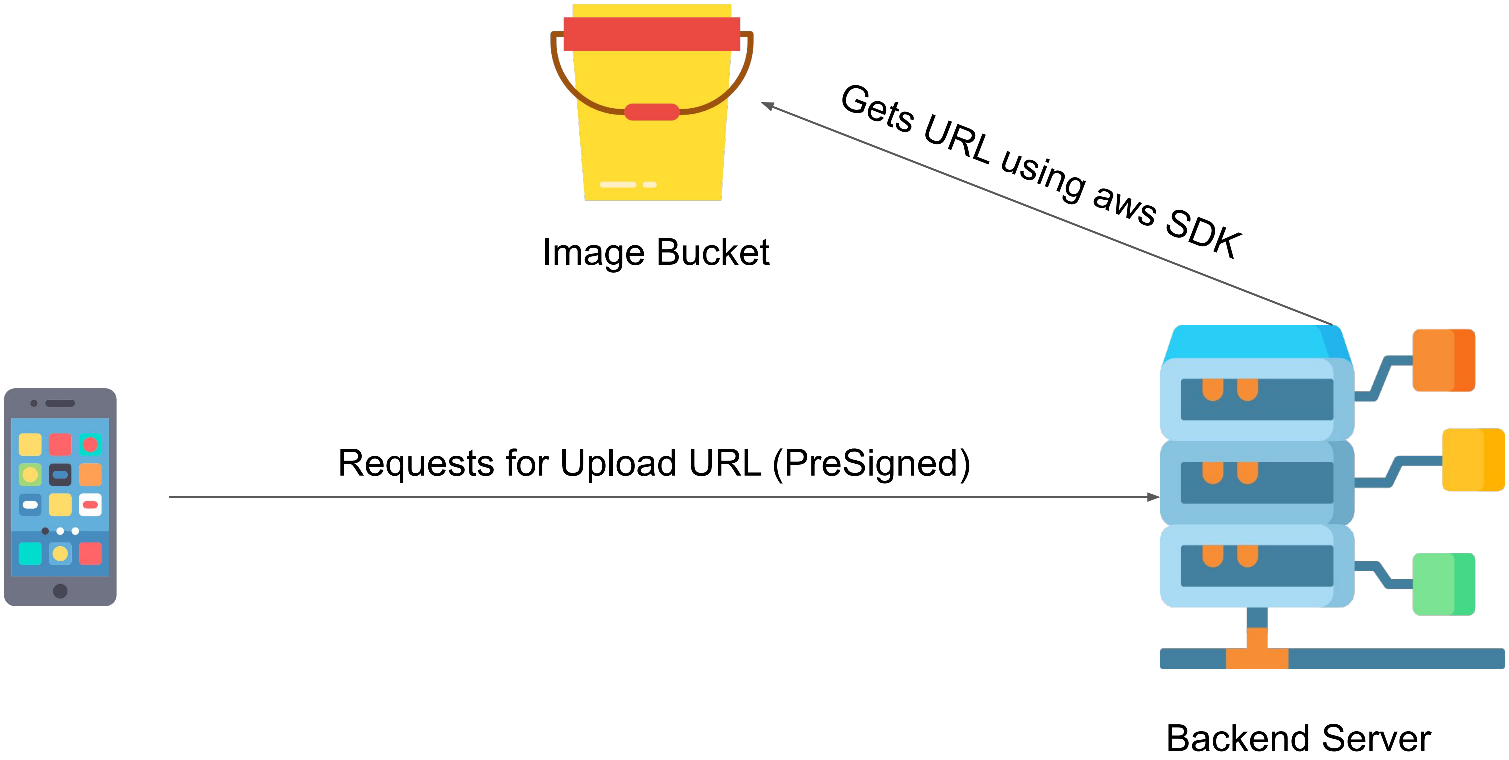
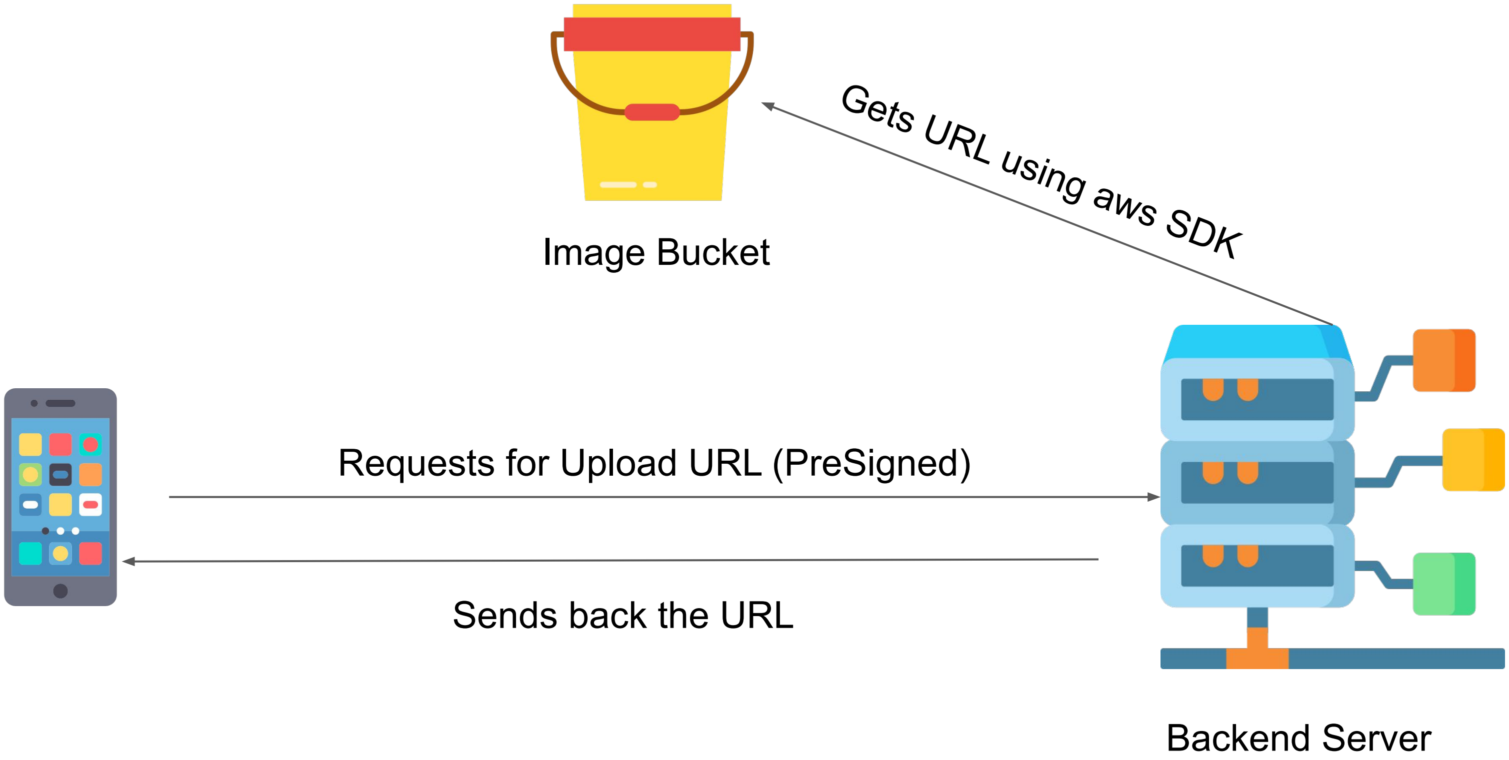
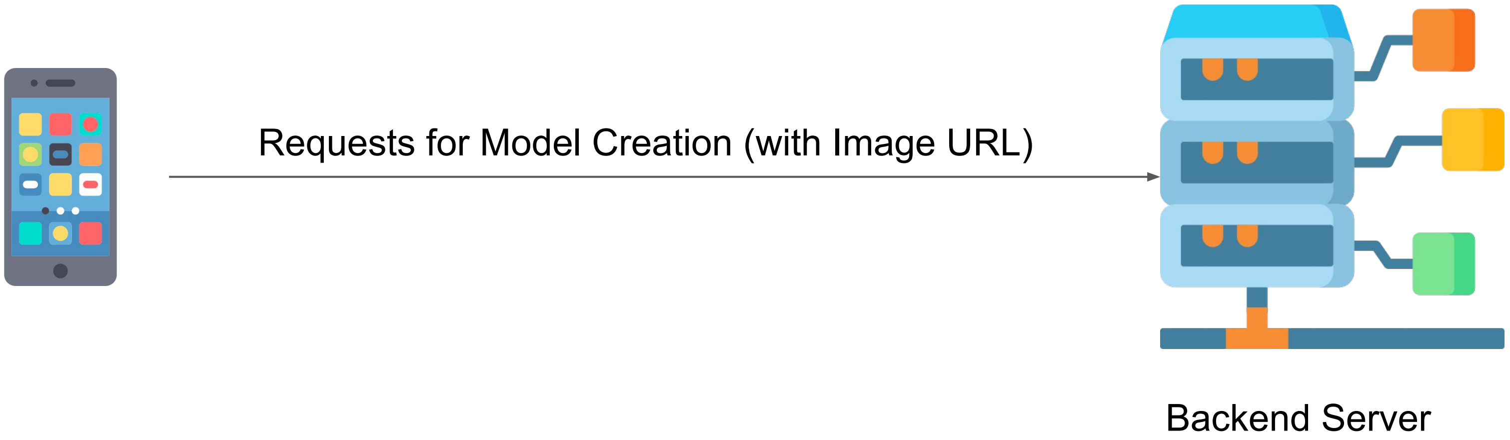


Image Bucket
Gets URL using aws SDK
Requests for Upload URL (PreSigned)
Backend Server

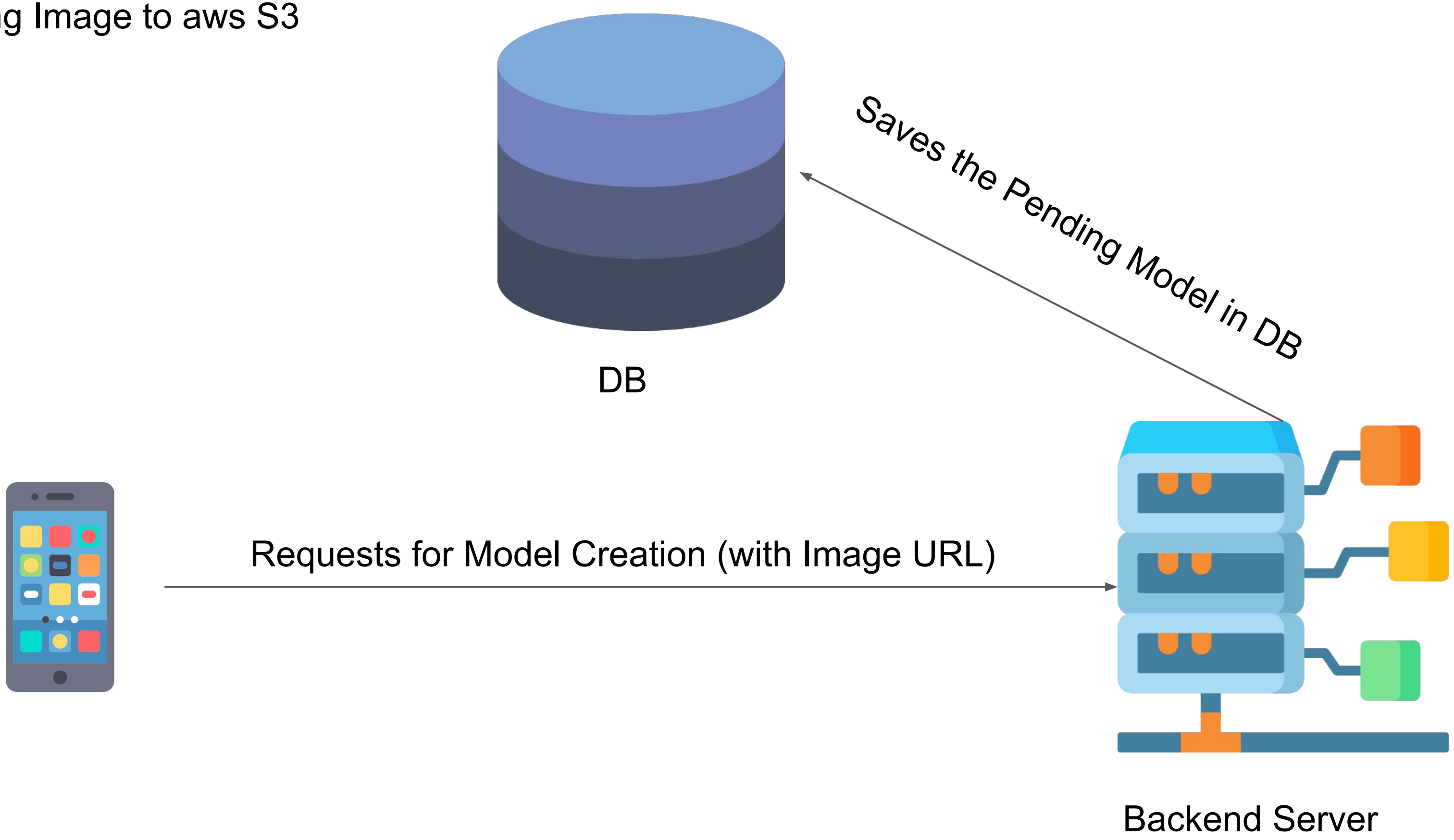
Uploading Image to aws S3



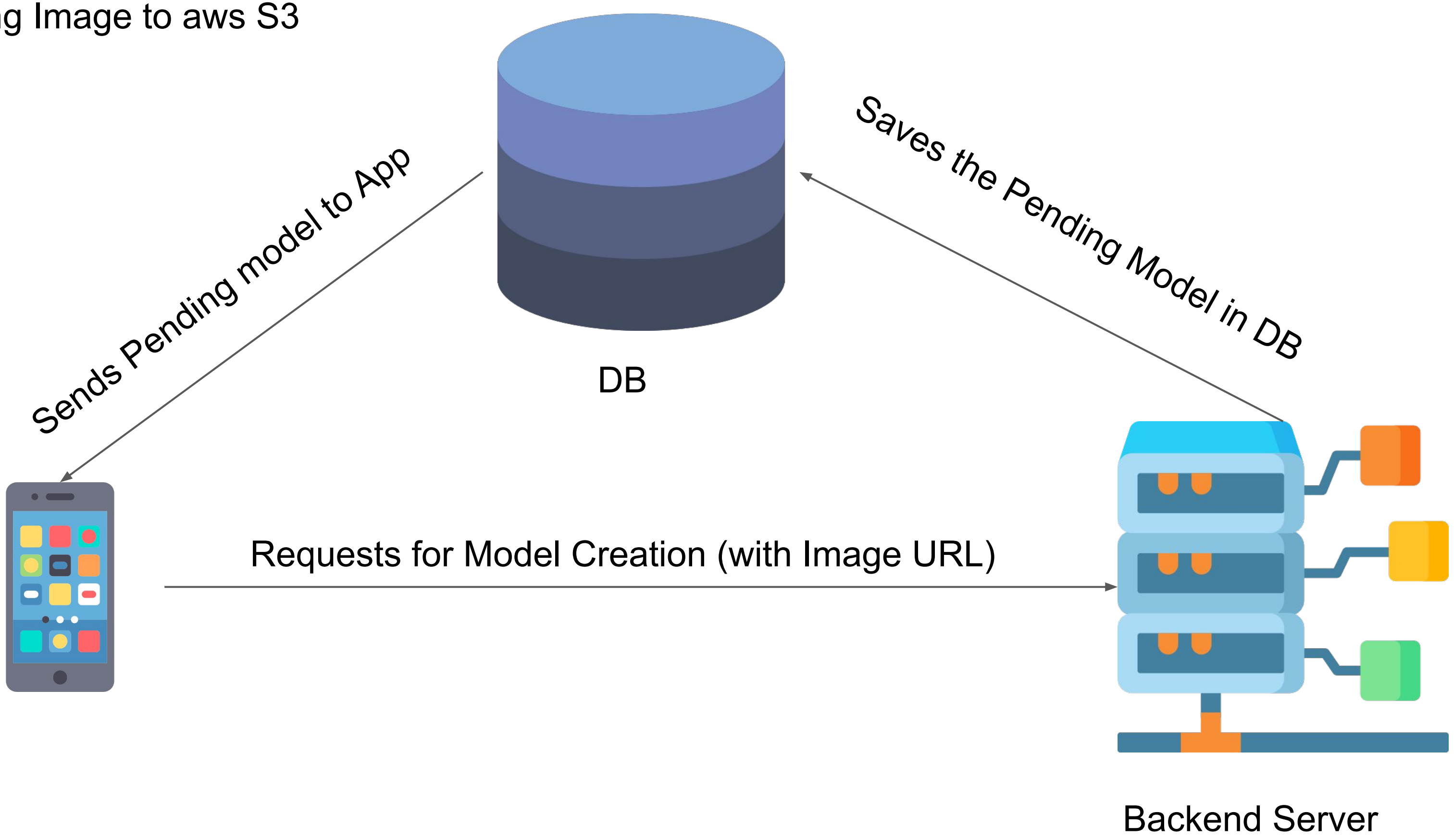
Uploading Image to aws S3

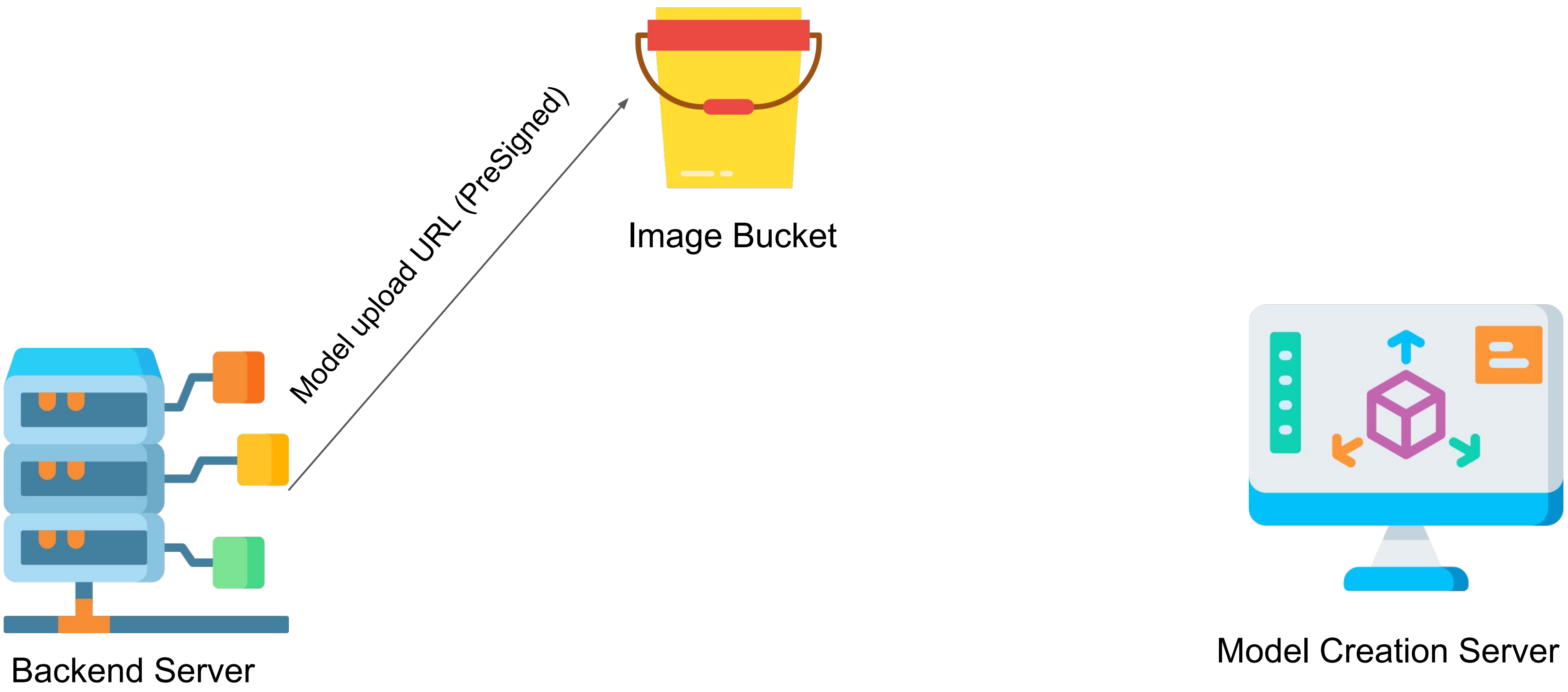


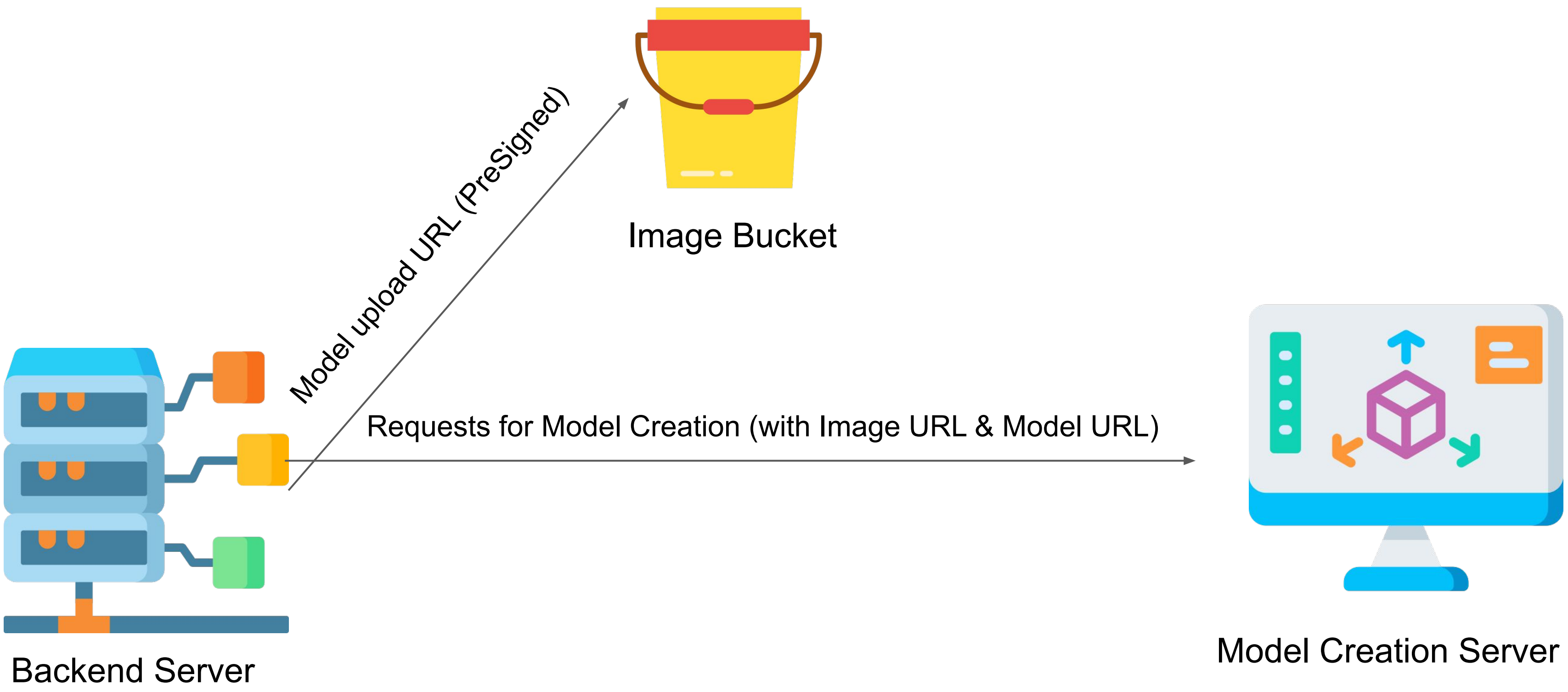
Uploading Image to aws S3



Uploading Image to aws S3




Model upload URL (PreSigned)
Image Bucket
Backend Server
Model Creation Server


Model upload URL (PreSigned)
Image Bucket
Requests for Model Creation (with Image URL & Model URL)
Backend Server
Model Creation Server

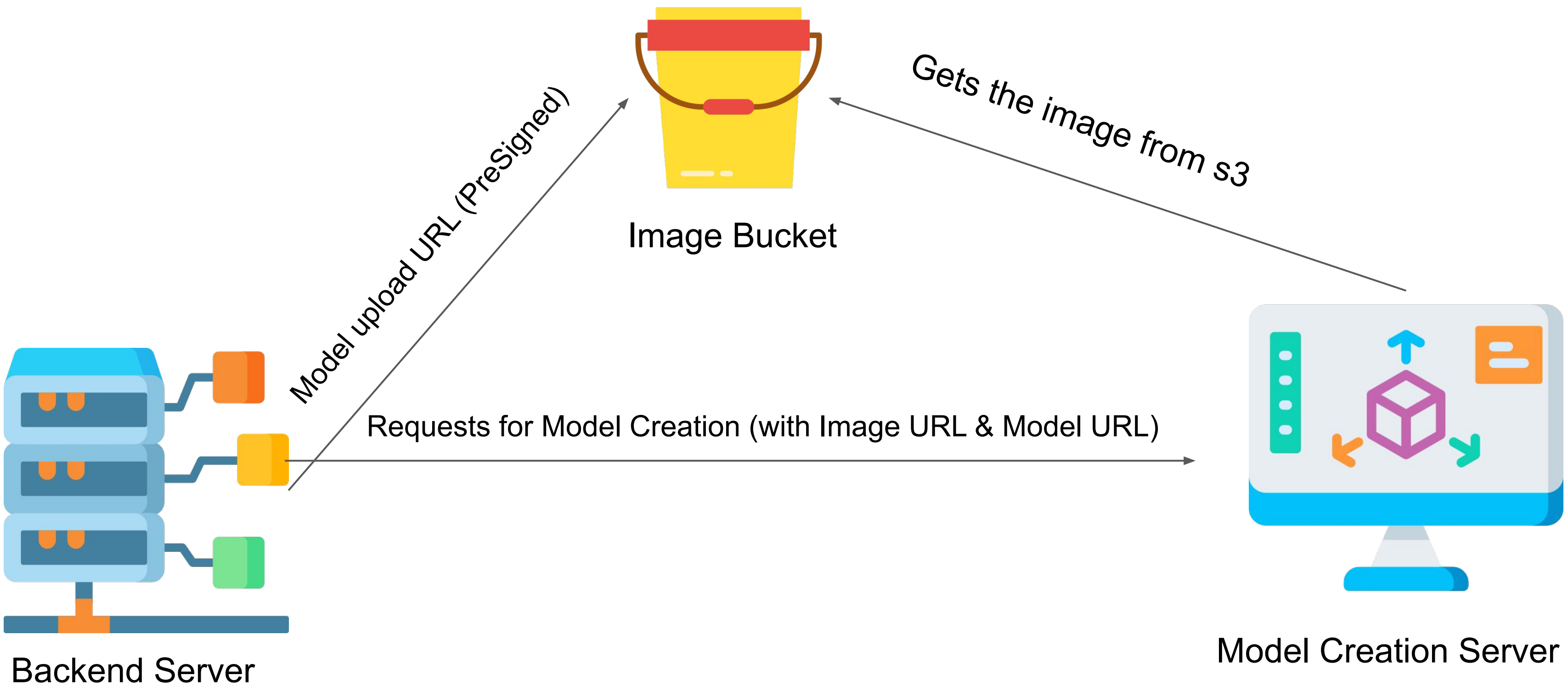


Image Bucket
Model upload URL (PreSigned)
Gets the image from s3
Requests for Model Creation (with Image URL & Model URL)
Backend Server
Model Creation Server

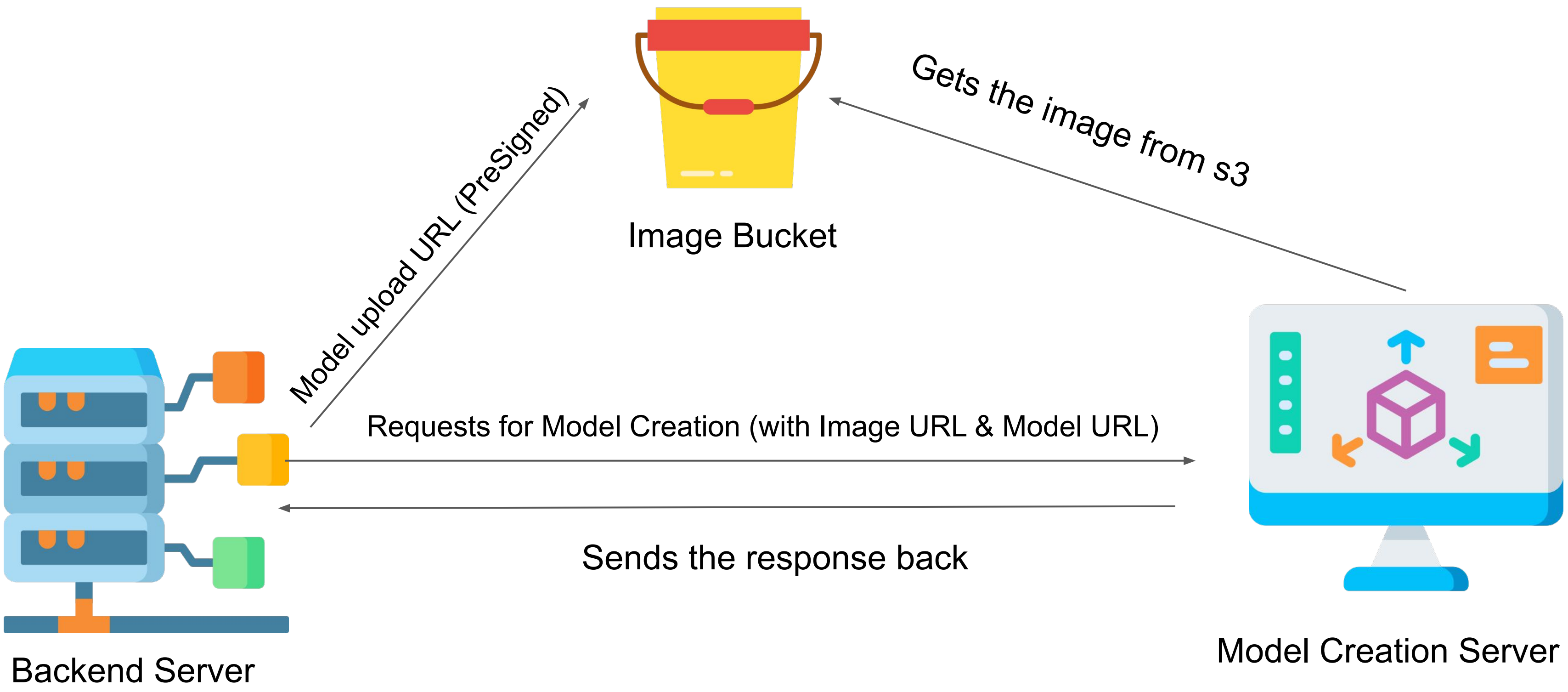


Image Bucket
Model upload URL (PreSigned)
Gets the image from s3
Requests for Model Creation (with Image URL & Model URL)
Sends the response back
Backend Server
Model Creation Server

Model Retrieval using QR code

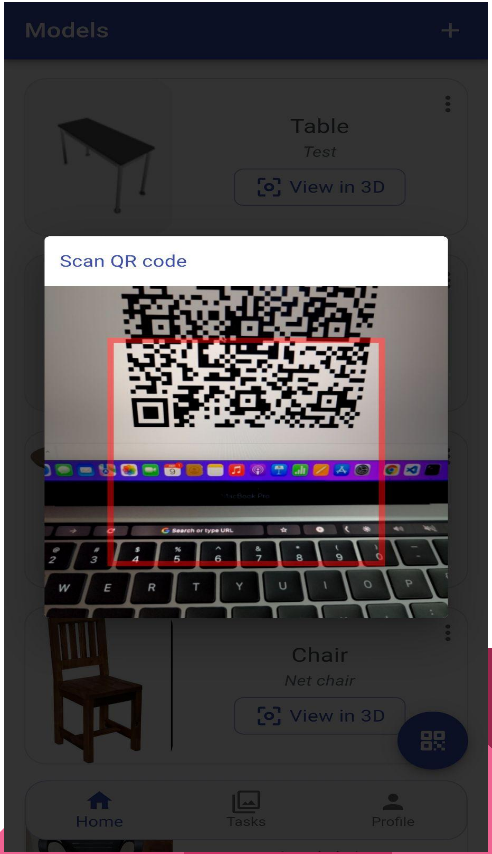
Every Processed 3D model in our Database has a QR code associated with it, which contains information such as model name and the url to download the model.

{
"model_url":"https://btp-model.s3.ap-south-1.amazonaws.com/b74681da17f97068c83346dbf17b0902",
"title":"Table"
}

Model Retrieval using QR code

The QR code can be viewed in App which can be printed, saved as pdf or can be shared via email, drive, whatsapp etc.

The app has a QR scanner which scans the generated QR and renders the models in real time.



Mobile App Tech Stack

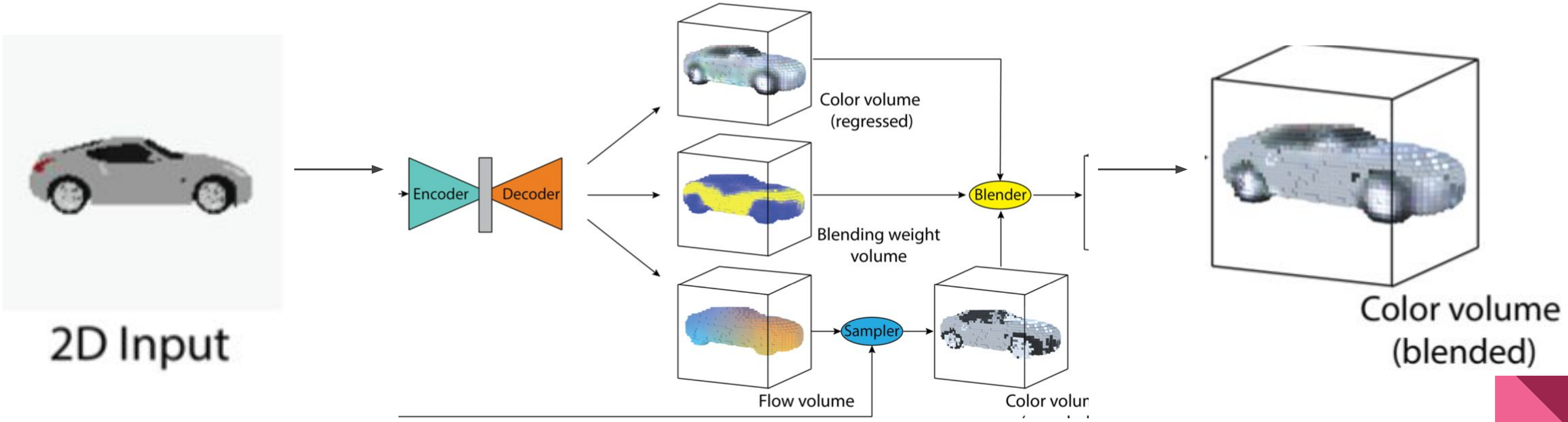
- ARCore
- Android
- NodeJS
- React
- AWS S3 bucket

Major Challenges We Faced

1. High Computation resources required for Color Learning

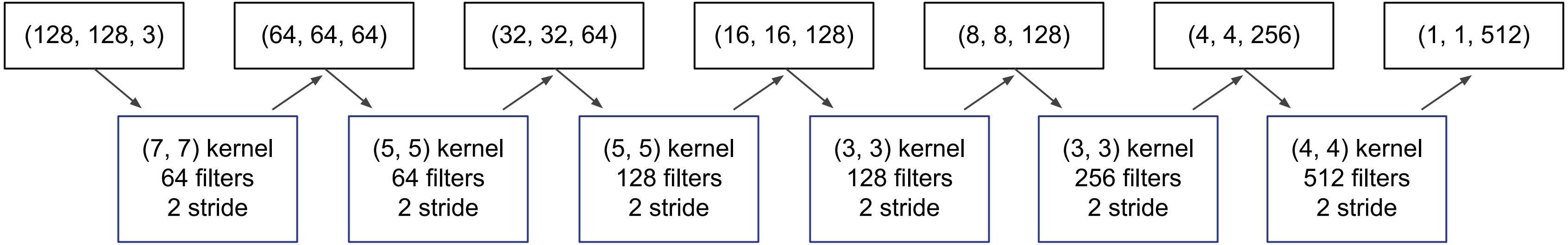
What is Color Learning

- For getting the colors, the idea is to build the same 3d structure. Each 3D voxel gets its shape from corresponding voxel index.
- Two combined strategies - sample color from 2d image and other regress colors directly.

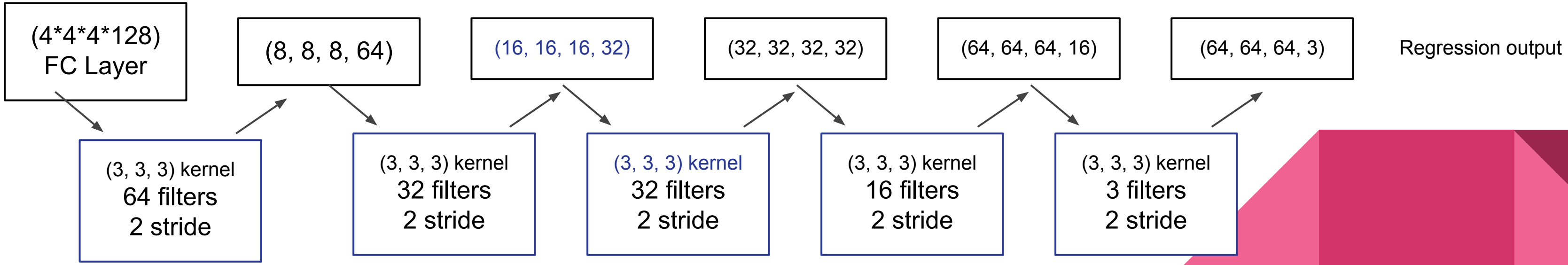


Color Network

Same Encoder - 6 2D convolution layers



Decoder - 5 3D convolution layers



Regression and blending

- We use another strategy of Regression to infer 3D surface colors. This will give better results for high rough regions.

Loss functions

- Target flow loss $$L_{flow} = \frac{1}{S}\sum_{i=1}^{S} \|V_i^{flow} - \hat{V}_i^{flow}\|_2$$
- Regression loss $$L_{clr\_regress} = \frac{1}{S}\sum_{i=1}^{S} \|V_i^{color} - \hat{V}_i^{clr\_regress}\|_2$$
- Weighted loss $$\hat{V}_i^{clr\_blend} = w \times \hat{V}_i^{clr\_sample} + (1-w) \times \hat{V}_i^{clr\_regress}$$
- Blend loss $$L_{blend} = \frac{1}{S}\sum_{i=1}^{S} \|V_i^{color} - \hat{V}_i^{clr\_blend}\|_2$$
- FInal loss $$L = L_{flow} + L_{clr\_regress} + L_{blend}$$

We only use Target flow during training.

We only train for colored voxels.

Complexity Explosion for color learning

- Number of images = 8.1k * 13 = 105k files
    - Only 10% extraction of files possible in drive in 13 hours
- Network * 4
- Model + First batch of data did not fit in google colab RAM.
- Training will dramatically slow down.
- We have the color learning architecture built up. We have documented the way to train the color learning if the interested person has enough compute resources to do so.

2. Finding unbiased metric to check model performance

Find a good metric

- We need a metric to decide whether the model's performance is good or not on a furniture category.
- Looking at Mean Squared Loss is not a good choice, because for complex images even with higher loss, the resulting 3D structure can be visually appealing.
- If humans sit in the process of labelling good/bad outputs from the model, it adds that person's bias in the validation process.
- Finding a good unbiased metric for our use case, is an open ended research question. If time permitted, we wanted to go back to literature search and tackled this problem.
- This problem arise at the time of we collecting different distribution statistics of furniture category and checking the model performance on them.

An Example Demo Output of the App

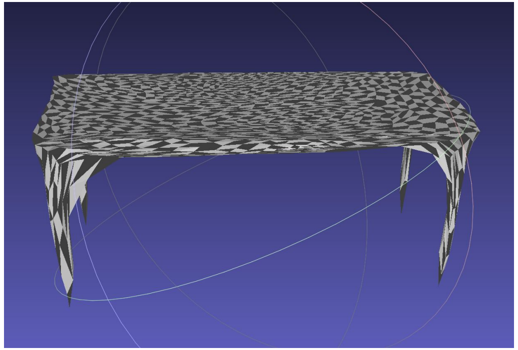


Table	→	Generated by CVN	→	In real world
Mobile App		Server		Mobile App

Results and Conclusion

- We proposed a data driven solution for our use case.
- We got visually appealing results for the furniture categories the model is trained on.
- We built a mobile app that integrates end-to-end from Users to Servers and then back to Users.
- Whatever challenges we faced, we have extensively documented them.
- The work done here can be further carried by solving the challenges.
- Finally, we want to thank our BTP advisor, Dr. Subu Kandaswamy for his guidance throughout BTP.

Contributions

- Sayam Kumar - S20180010158
  - Model building related tasks
    - Preparation of Data Pipelines
    - Implementation of Model Architecture
    - Model Training and Model Deployment
- Nitin Kumar Chauhan - S20180010119
  - Mobile App related tasks
    - Model Rendering using ARCore
    - Database and Cloud storage management
- Dasari Jayasree - S20180010047
  - Literature Review

References

- Tensorflow docs https://www.tensorflow.org/api_docs
- AR Core docs https://developers.google.com/ar/reference
- Numpy docs https://numpy.org/doc/
- Scipy docs https://www.scipy.org/docs.html
- Im2Avatar paper https://arxiv.org/pdf/1804.06375.pdf
- Wikipedia https://en.wikipedia.org/wiki/Main_Page