Transfer Learning for High Quality Monocular Depth Estimation

Raghav Arneja
College of Electrical Engineering and Computer Science
Queen Mary University
United Kingdom, London
Email: ec20059@qmul.ac.uk

***Abstract*—Depth estimation from a single 2D image is becoming a popular problem because of its multiple applications which include robot vision, 2D to 3D image/video conversion and self-driving cars. It is also a fundamental task when it comes to scene understanding and reconstruction. With the use of transfer learning, this study proposes numerous convolutional neural networks for constructing a high-resolution depth map given a single RGB image. Also, we will follow standard encoder decoder architecture where the features will be extracted using high-performing pre-trained networks when we initialize our encoder to get state of our results. Our main focus is to compare the effects of various pre-trained networks when it comes to depth estimation.**

***Index Items* - Transfer Learning, Depth estimation, Encoder-decoder, single view image reconstruction.**

I. Introduction

When the conventional 2D cameras takes pictures or captures a shot, the depth information gets lost which gives us a 2D image with a projection onto the 2D plane. So, estimating this lost dimension and recovering the lost depth has become more important in these years. Navigation and scene understanding, augmented reality [1], image refocusing [2], and segmentation [3] can all benefit from having a comprehensive depth map of the real environment. Recent studies have focused mostly on using CNNs and other deep learning methods for the reconstruction of images from 2D to 3D. The most widely used methods for depth estimation are the Shape From X method for obtaining the scene depth shape from image brightness and shade, different perspectives, luminosity and texture information. Although numerous technologies can directly measure depth, the technology is costly. Binocular depth estimation can also be employed, although it must use stereo matching for pixel correspondence and parallax calculation when dealing with binocular pictures. Despite the fact that monocular depth estimation [4–7] is an ill-posed issue for a single image lacking the requisite geometric information, it has received a lot of attention because it is comparatively less expensive and easier to popularise. With the development in technology and computer vision, monocular depth based on deep learning has made great progress. Eigen et al. [8] proved

through experiments that a Convolutional Neural Network can be used to determine the image's relative depth (CNN).

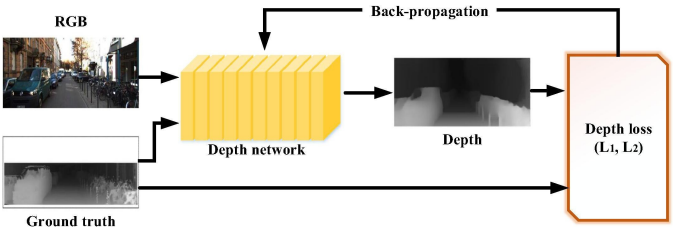

Fig. 1. The general supervised learning model for monocular depth estimation, with RGB and GT depth images as inputs and the estimated depth map as output.

(CNNs) has accelerated the research of depth estimation [8–12]. Though use of CNN's have shown steady improvements in the dense depth estimation strategies, there are several scopes of improvements in image resolution and quality of predictions in the resulting depth maps. MDE's progress is slow as compared to depth estimates from stereo pictures or video sequences, where significant improvement has been accomplished [13, 14]. MDE is an ill-posed problem since it allows an infinite number of different 3D scenes to be combined into a single 2D image.

We will focus on an efficient deep neural network architecture for computer vision in this study, which takes its name from Lin et al Network in Network paper [8] and the famous "we need to go deeper" internet meme. In our case, the word "deep" is used in the sense of increased network depth.The larger the network, the more parameters it has, which makes it more prone to overfitting, especially if the number of labelled samples in the training set is restricted. The consumption of computational resources is greatly increased when network size is uniformly raised. The introduction of sparsity and the replacement of fully connected layers with sparse layers, even inside convolutions, would be a basic way to solve both of these concerns.

Contributions: We make a two-fold contribution. First, we offer all transfer-learning based network designs that will

produce more accurate and high-quality depth estimates. With fewer training iterations and parameters, the resulting depth maps will capture object boundaries more accurately. Second, to attain state-of-the-art results and faster learning, we will create a corresponding loss function, data augmentation,

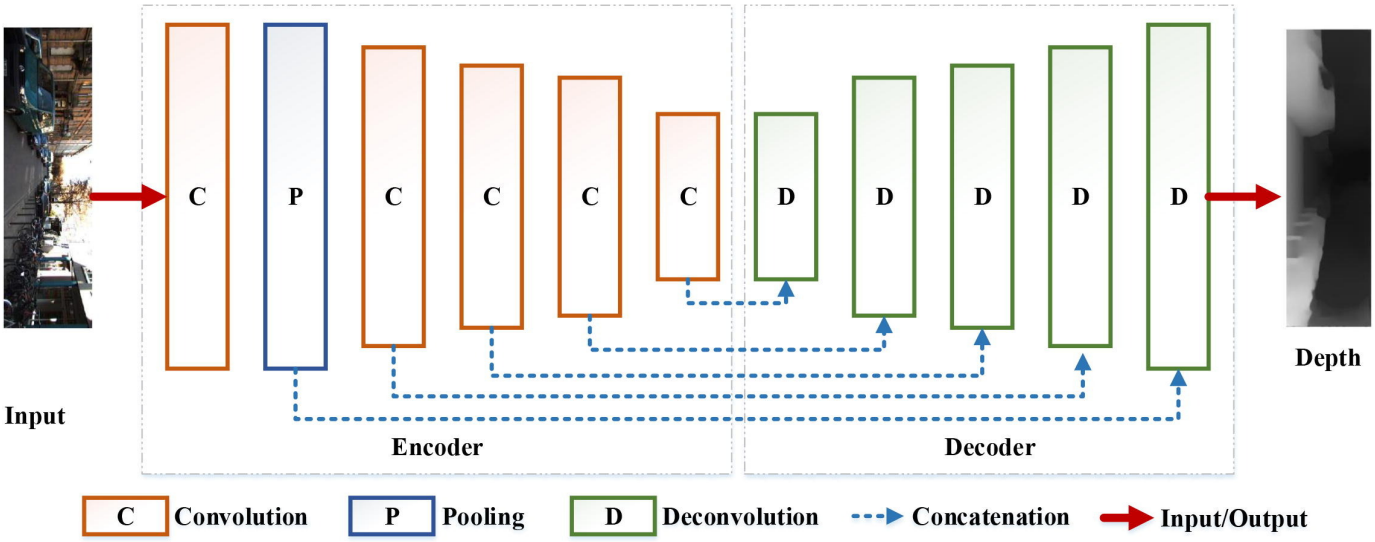

Fig. 2. The general pipeline of deep learning for monocular depth estimation. The encoder network on the left learns depth features layer by layer, while the decoder network on the right recovers the depth map.

and tuning hyper parameters or changing learning algorithms.

II. Section Example

According to the many generalisations of this study, we summarize the past research on Depth estimation and also discuss various deep learning models For vision and image processing applications, that learn general shapes of high-dimensional data.

A. Depth estimation

Many CNN approaches have considered monocular depth estimation problem as a regression of the depth map from a single RGB image [4, 5, 8, 15]. Understanding the 3D structure of situations from 2D photos requires depth estimation. Early research focused on estimating depth from stereo pictures using geometry-based algorithms [16–18] that leverage point correspondences between images and triangulation to estimate depth. Ground breaking work Saxena et al. [6] who used linear regression and an MRF to predict depth from a set of picture attributes, eventually expanded their work into Make3D [19] system modelling for 3D image production. Liu et al. [20]. suggest depth estimate from projected semantic labels after that. For depth estimate from single photos, there are semi-automatic and automatic approaches. Horry et al [21] proposes an interactive trip within the picture, in which the user adds planes to an image to create motion. In recent years, Automatic methods came up for single depth estimation like Hoiem et al. [22] proposes an automatic photo pop-up that reconstructs an outdoor image using the image's assumed planar surfaces. Delage et al. [23] create a Baysian framework for indoor scenarios. Depth Transfer, invented by Karsch et al. [24], is a common depth estimation method. This approach first creates a large-scale RGBD picture and feature database, then transfers the depth of multiple comparable images after warping and optimising operations to obtain the depth of the input image. Besides this, There are also various more initiatives to estimate depth using unified global and local data. To accomplish inference through a graphical model, Liu et [25] use discrete variables to represent relationships between surrounding superpixels and continuous variables to encode the depth of the superpixels in the input image. Eigen et al. [8] presented a multi-scale network consisting of coarse and fine scale networks. Despite the fact that the goal of a multi-scale network is to retain details while resolving global information, the fine network's early pooling and striding causes it to lose a lot of information. Furthermore, unlike newly discovered densely linked structures that can keep information while passing features, the fine network loses more details after repeated convolution layers. Depth estimate was viewed by Fu et al. [5] as a multi-class classification problem. To gather global information, they employed a dense feature extractor followed by a scene understanding module to broaden the field of view. The lack of connections between layers, on the other hand, results in a lack of detail in the output. Alhashim et al [7] used a transfer learning technique to train a simple encoder-decoder design. After this, a lot of approaches have been introduced following this concept [26–31].

B. Artificial Neural Networks

Artificial neurons (ANNs) are a class of Machine Learning algorithms whose architecture is built on a set of connected units or nodes known as artificial neurons that loosely replicate the neuron in the brain. It is shown to be powerful predictive tool in disciplines including computer vision and natural language processing [32]. ANNs are made up of layers of simple computing units (nodes), with each node's output being an elementwise combination of their inputs after they've

been processed through a non-linear activation function. [33].

C. Transfer Learning and Deep Networks

Transfer learning has been shown to be quite successful in a variety of situations. Zamir et al. [34] have researched and modelled the transfer learning dependency of 26 tasks, 16 of which are connected to 3D or geometric concepts. Transferring the paradigm for object classification to depth estimation was particularly effective when Alhashim et al. [7] brought this notion to the problem of depth estimation.

Simple recognition tasks, especially when augmented with label-preserving transformations, may be solved pretty successfully with datasets of this size. On the MNIST digit-recognition challenge, for example, the current best mistake rate (0.3 percent) approaches human performance [35]. However, due to the enormous complexity of the object recognition challenge, even a dataset as large as ImageNet is insufficient to specify this problem, thus our model needs include a lot of previous knowledge to compensate for the data we don't have [36]. CNNs feature far fewer connections and parameters than typical feed forward neural networks with similar-sized layers, making them easier to train while their theoretically-best performance is expected to be just slightly inferior [32]. In large-scale image identification tasks, the shallow neural network approach has some limitations. Simonyan and Zisserman [37] proposed the VGG to further investigate the performance of the deeper network model. The fundamental contribution of VGG is a detailed examination of networks of increasing depth utilising an architecture with very small (3 x 3) convolution filters, which reveals that expanding the depth to 16 to 19 weight layers achieves a considerable improvement over prior-art configurations. Lin et al presented Network-in-Network as a method for increasing the representational ability of neural networks. The GLM is replaced in NIN by a "micro network" structure, which is a nonlinear function approximator in general. The NIN's overall structure is made up of numerous mlpconv layers stacked on top of one other. Network-in-Network is an approach proposed by Lin et al. [38] in order to increase the representational power of neural networks.The GLM is replaced in NIN with a micro network topology that approximates general nonlinear functions. The ILSVRC-2014 image classification competition was won by GoogLeNet. The inception module was effectively proposed by GoogLeNet, who was motivated by network in network to widen the network structure. The problem of disappearing gradients has been largely addressed by normalised initialization and intermediate normalisation layers, according to He et al [39]. The accuracy of the training set was saturated or even reduced as the number of network layers increased. Overfitting should be better in the training set, thus this cannot be regarded as overfitting. To address the aforementioned issues, he et al. proposed the ResNet. ResNet's key contribution is to eliminate the negative consequences (degradation) of growing network depth, allowing network performance to be enhanced merely by increasing network depth. The Residual Network design, ResNet [40], was named the ILSVRC 2015 winner. Kaiming was the one who came up with Resnet. He set out to create ultra-deep networks that were free of the vanishing gradient problem that plagued predecessors.ResNet is built with a variety of layer counts: 34, 50, 101, 152, and even 1202. ResNet is a residual feed forward network with a regular feed forward network.The residual unit's final output is x_l, which can be calculated using the following equation:

$$x_l = \mathcal{F}\left(x_{l-1}\right) + x_{l-1}$$

Zagoruvko el proposed a broader version of residual network in 2016 [41]. In 2016, [42] the aggregated residual transformation approach was suggested as an improved residual network approach. Inception-architecture v4 [43] introduces the Inception Block with Residual Connections concept. After this, PolyNet, an upgraded version of the Inception-Residual network, was recently proposed [44]. WideResnet and ResNext are two well-known ResNet variations that investigate the dimensions of width and cardinality, respectively. Resnet performance decreases consecutively whenever the network becomes extremely deep. By re-ordering activations in the ResNet module, Preactivation-ResNet [39] was able to solve this problem. Only extremely deep networks showed a performance increase of preactivationResNet over original ResNet [39]. ResNext looked into the effect of cardinality, which was found to be more successful than increasing depth or width. Instead than using the identity shortcut, DenseNet [45] effectively use the concatenation technique. Furthermore, CondenseNet [46], a version of DenseNet, takes the power of dense connections to a new level. Layers with differing resolution featuremaps are densely connected, which is one of the most notable distinctions from DenseNet. Layers with differing resolution featuremaps are densely connected, which is one of the most notable distinctions from DenseNet. Furthermore, dual path networks and mixed link networks use both identity shortcut and dense concatenation to combine ResNet and DenseNet into a single network.

As a result, our model should have a lot of prior knowledge to compensate for the lack of data [36].

III. Problem Statement

Scene depth estimation is significant in computer vision because it improves the perception and understanding of genuine three-dimensional scenes, which is useful in a variety of applications like robotic navigation, autonomous driving, and virtual reality. Binocular depth estimation can also be employed, although it must use stereo matching for pixel correspondence and parallax calculation when dealing with binocular pictures. The computational complexity is high when working with binocular images and the matching effect is not good for low texture scene. Monocular depth estimation has been widely concerned as it's relatively cheaper and easier to popularize. The state of the art [7] uses transfer learning techniques for the encoder, using

Densenet-169 [45] network pretrained on ImageNet [47]. One of the problem seen is that they used very Deep networks as encoders and never used more compact networks. Building on previous work, this projects aims on using more compact, not so deep networks as encoders pretrained on ImageNet and then evaluate the results, compare them with the state of art methods and get results on test data. When using deep networks we will also use MobilenetV2 [48]. It was designed to drastically reduce the network's complexity cost and model size, making it appropriate for mobile devices and other devices with limited processing power.

IV. Proposed Method

In this section, we basically describe the methods for estimating depth maps on RGB images. We will describe the architectures where our encoders we use pretrained architectures like densenet161, resnet etc. all pretrained on ImageNet [47]. This vector is then passed into a succession of up-sampling layers, resulting in a final depth map with a resolution half that of the input. Our decoder is made up of these upsampling layers [49] and their related skip-connections. Batch Normalization [50] or other sophisticated layers proposed in recent state-of-the-art approaches [5] are not included in our decoder. The complexity of both the encoder and decoder, as well as its relationship to performance, is then discussed. Then, for the particular task, we offer a suitable loss function. Finally, we discuss effective augmentation policies that greatly aid the training process.

A. Network Architecture

Densenet: In a Standard Convolutional Neural Network, we start with an input image, which is then transmitted through the network to produce an output predicted label in a straightforward manner.

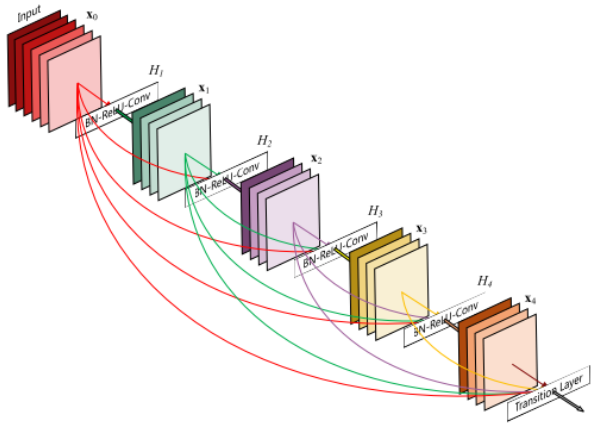

Fig. 3. Each Densenet layer is connected to all the other layer

Except for the first convolutional layer (which receives the input image), each convolutional layer takes the output of the previous convolutional layer and produces an output feature map, which is then sent to the next convolutional layer. There are L direct connections between each layer and its subsequent layer for L layers. The DenseNet [45] architecture is all about modifying this standard CNN architecture. DenseNet gets its name from the fact that each layer in a DenseNet architecture is linked to the other layers. L layers have $L(L+1)/2$ direct connections. Each layer uses the feature maps of all previous levels as inputs, and its own feature maps as inputs for subsequent layers. During training, the input to our ConvNets is expected to be at least 224×224 RGB. Traditional feed-forward networks connect the output of the L^{th} layer to the $L+1^{th}$ layer, as we know. And the skip connection can be represented as:

$$X_l = H_l(X_{l-1}) + X_{l-1}$$

In DenseNet architecture, the dense connectivity can be represented as:

$$X_l = H_l([X_0, X_1, ..., X_{l-1}])$$

MobilenetV2: In mobilenetv2 [48], there are two different kinds of blocks. One is a stride 1 residual block. Another option for shrinking is a block with stride of 2. For both types of blocks we have 3 layers.

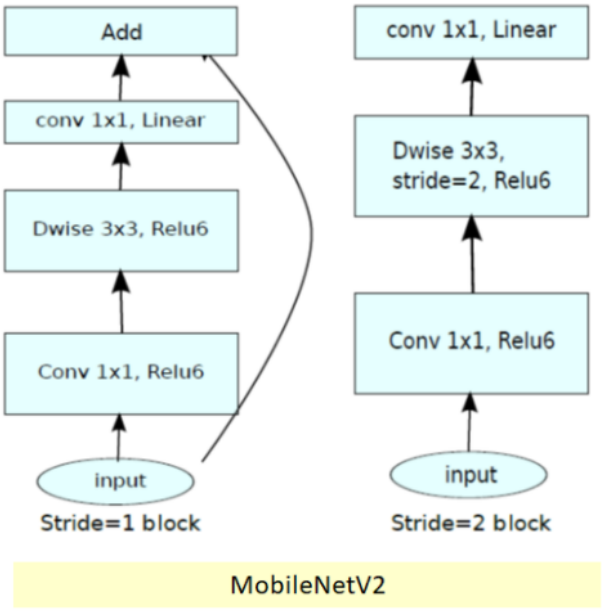

Fig. 4. MobilenetV2 Convolutional Block

Unlike, mobilenetv1, the first layer is 1x1 convolution with ReLU6. The second layer is depthwise convolution. In the third layer, another convolution is employed, but this time there is no non-linearity. According to the statement, deep networks will only have the power of a linear classifier on the non-zero volume part of the output domain if

ReLU is performed again. There's also a t expansion factor to consider. $t = 6$ was employed in all of the primary experiments.The internal output would have 64 x t= 64 x 6 = 384 channels if the input had 64 channels.

ResNet: ResNet's contribution [40] to alleviating the degradation problem was the incorporation of residual learning for the training of very deep learning models. As the name suggests, the new objective is to learn an alternative underlying residual mapping.

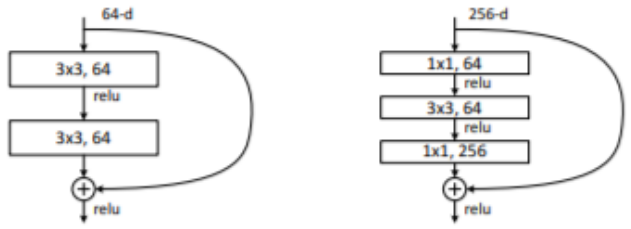

Fig. 5. F is a more detailed residual function for ImageNet. A building block (on 56 x 56 feature maps) on the left. Right: a ResNet-50/101/152 "bottleneck" building block.

The majority of network convolution layers use 3x3 filters and follow two simple design principles: first is for the same output feature map size, the layers must have the same number of filters; and second if the feature map size is halved, the number of filters must be doubled to maintain the time complexity per layer. To do direct downsampling, we employ convolutional layers with a stride of 2. The network is completed with a global average pooling layer and a 1000-way fully-connected layer with softmax. There are a total of 34 weighted layers. We create shortcut connections based on the plain network that transform the network into its residual version. If the input and output dimensions are the same, the identity shortcuts can be employed right immediately. The incorporation of residual block in the ResNet design combats the vanishing gradient problem. The skip link between the layers adds the preceding layers' output to the stacked layers' output.

B. Complexity and performance

We have experimented with many deep learning architectures for our encoders which includes deeper network like Desnenet161 and Resnet50 .What we discovered empirically is that, in the context of a depth estimation encoder-decoder architecture, recent trends toward more complicated convolutional blocks do not necessarily improve performance. Our tests show that a simple decoder [51] consisting of a two-step bilinear upsampling followed by two typical convolutional layers functions admirably. These observations can become a part of future work where we deeply investigate how complicated convolutional blocks can be a part of our encoder-decoder architecture.

C. Loss function

The difference between the groundtruth depth map y and the prediction of the depth regression network $\hat{y}$ is considered by a standard loss function for depth regression problems [8]. In our method, We're looking for a loss function that strikes a compromise between recreating depth images by minimising the difference in depth values while simultaneously punishing distortions of high-frequency details in the depth map's image domain. These features usually match to the scene's object borders. We define the loss L between y and $\hat{y}$ as the weighted sum of three loss functions for training our network:

$$L(y, \hat{y}) = \lambda L_{\text{depth}}\,(y, \hat{y}) + L_{\text{grad}}\,(y, \hat{y}) + L_{SSIM}(y, \hat{y})$$

the first loss term L_{depth} is the point-wise L1 loss defined on the depth values:

$$L_{\text{depth}}\,(y, \hat{y}) = \frac{1}{n}\sum_{p}^{n} |y_p - \hat{y}_p|$$

The Second Loss term L_{grad} is the L1 loss defined over the image gradient g of the depth image:

$$L_{\text{grad}}\,(y, \hat{y}) = \frac{1}{n}\sum_{p}^{n} \left|\boldsymbol{g}_{\mathbf{x}}\,(y_p, \hat{y}_p)\right| + \left|\boldsymbol{g}_{\mathbf{y}}\,(y_p, \hat{y}_p)\right|$$

where g_x and g_y compute the differences in the x and y components for y and $\hat{y}$ depth image gradients, respectively.

Finally, L_{SSIM} employs the term Structural Similarity (SSIM), which is a widely used metric in image reconstruction. It has recently been demonstrated to be a good loss term for CNN depth estimation. We define loss L_{SSIM} as follows:

$$L_{SSIM}(y, \hat{y}) = \frac{1 - \text{SSIM}(y, \hat{y})}{2}$$

For the loss term L_{depth}, note that we only define one weight parameter λ. We arrived at the value of λ = 0.1 as a fair weight for this phrase by trial and error.

D. Augmentation policy

In data analysis, Data augmentation refers to methods for enhancing the amount of data available by adding slightly modified copies of existing data or synthesising new data from existing data. When training a machine learning model, it functions as a regularizer and helps to reduce overfitting [32]. Since our network is designed to estimate depth maps of an entire image, not all transformations would be appropriate as some will take out the important aspect of the ground truth image and we will not have a good interpretation of that image. For now the most relevant transformation that can be seen is mirroring i.e horizontal flipping of the images at a probability of 0.5. Applying a vertical flip to an image of an indoor scene may not aid in the understanding

of statistical qualities that are predicted (e.g. geometry of the floors and ceilings). Finding better data augmentation policies and their probability values for the problem of depth estimation is a promising area of research for the future [52].

E. Optimizer

Adam [53, 54], an approach for efficient stochastic optimization requiring only first-order gradients and little memory. Adam is designed to combine the advantages of two recently popular methods: AdaGrad [55], which works well with sparse gradients, and RMSProp. Using exponential moving average, and corrects its bias the approach determines the gradient's 1st-order moment (the gradient mean) and 2nd-order moment (element-wise squared gradient). Learning rate times 1st-order moment divided by the square root of 2nd-order moment yields the final weight update. Adam uses three hyperparameters: learning rate, first-order moment decay rate, and second-order moment decay rate. As follows, we compute the decaying averages of past and past squared gradients, m_t and v_t, respectively:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2$$

The method's name comes from m_t and v_t which are the estimations of the first (mean) and second (uncentered variance) moments of the gradients, respectively. They account for these biases by computing bias-corrected first and second moment estimates, which are computed as follows:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

V. Experimental Results

A. Dataset

NYU-Depth V2 [56] dataset contains of video sequences of various indoor scenes recorded by Microsoft Kinect's RGB and Depth cameras. It features 1449 annotated RGB images and depth maps from 3 cities 464 scenes, 407024 unlabeled pictures. Each object is labeled with a class and an instance number. The dataset has several components:

- Labeled: A subset of the video data accompanied by dense multi-class labels. This data has also been preprocessed to fill in missing depth labels.
- Raw: The raw RGB, depth and accelerometer data as provided by the kinect.
- Toolbox: Useful functions for manipulating the data and the Labels.

We train our methods on a 50K subset of this dataset. All our network produces images at a resolution of 320 x 240. We use the original resolution of the input photos

Fig. 6. Indoor Images loaded from the NYU-Depth V2 Dataset. Top are original RGB images and there corresponding depth maps in the bottom.

for training and downsample the ground truth depths to 320 x 240 pixels. During testing, we compute the whole test image's depth map prediction and then upsample it by 2x to match the ground truth resolution. We compute the final output at test time by averaging the prediction of an image and the prediction of its mirror counterpart.

B. Implementation details

The implementation is done in Pytorch [57] and the code is trained on google colab pro that offers Tesla P100 GPU with 25GB Memory. Our encoder is trained with various Deep architectures like ResNet, DenseNet etc. all pretrained n ImageNet [47]. The weights of the decoder are randomly initialized [58]. We utilised the ADAM optimizer [53] in all of the tests, with a learning rate of 0.0001 and parameter values of β_1 = 0.9 β_2 = 0.999. The batch size is set to 7 for everyone. NYU Depth V2 requires a million iterations of training, which takes 18-20 hours to complete.The total number of Trainable parameter were around 42M for all the networks.

C. Evaluation

we define a few error metrics and we use these metrics to compare our methods with each other and with the state of the art method.

- average relative error (rel): $L_{SSIM}(y, \hat{y}) = \frac{1-\mathrm{SSIM}(y,\hat{y})}{2}$
- root mean sqaured error (rms): $\sqrt{\frac{1}{n}\sum_p^n \left(y_p - \hat{y}_p\right)^2}$
- average ($\log_{10}$) error: $\frac{1}{n}\sum_p^n \left|\log_{10}\left(y_p\right) - \log_{10}\left(\hat{y}_p\right)\right|$;
- threshold accuracy (δ_i): % of y_p s.t. $\max\left(\frac{y_P}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}\right) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3$

where y_p represents a pixel in the depth image y, $\hat{y}_p$ represents a pixel in the anticipated depth image $\hat{y}$, and n represents the total number of pixels in each depth image.

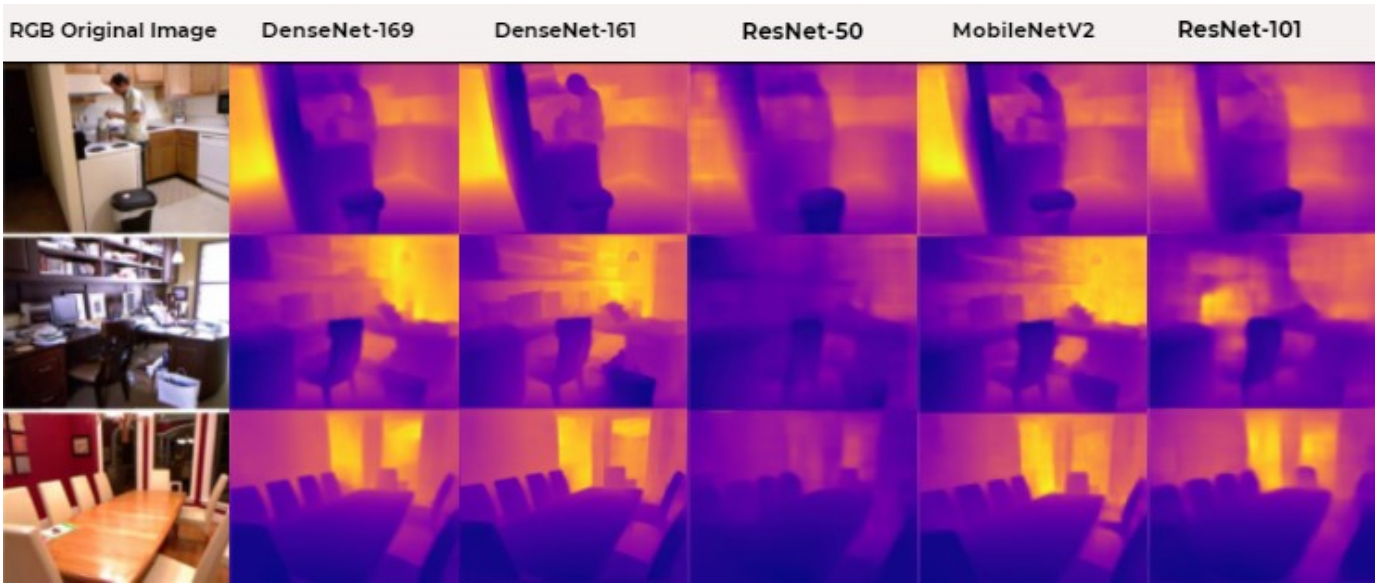

Fig. 7. These are Depth predictions from our various Encoder models. The very first column is the RGB Test Images. Second column is the State of the art predictions from DenseNet-169 [7]. Third Column is the Predictions from Densenet-161. fourth column is ResNet-50, then we have Predictions from MobileNetV2. last we have predictions from ResNet-101

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	rel $\downarrow$	rms $\downarrow$	$log_{10} \downarrow$
DenseNet-169 [7]	**0.837**	0.967	0.992	**0.133**	**0.481**	**0.056**
Densenet-161	0.827	**0.969**	**0.993**	**0.133**	0.483	**0.056**
ResNet-50	0.664	0.900	0.971	0.209	0.701	0.086
MobileNetV2	0.792	0.955	0.990	0.146	0.525	0.063
ResNet101	0.696	0.912	0.973	0.197	0.663	0.081

TABLE I
COMPARISONS OF DIFFERENT METHODS ON THE NYU DEPTH V2 DATASET TRAINED IN THIS RESEACH. THE TABLE SHOWS RESULTS OF DIFFERENT ENCODER USED AND COMPAIRING THE RESULTS OF ALL THE METHODS PERFORMED

D. Comparing performance

All the models when trained took around 20 hours to finish as we had 50K training samples for each of the model. We can see the results in Table 1. We again trained the state of the art method [7] which was an encoder with DenseNet-169. We achieved similar results on our machine too. Then we tried DenseNet-161, although both the DenseNet performs the same but DenseNet-161 shows better 1-crop error rates on ImageNet [47] Dataset than DenseNet-169. Also for using more compact models we used ResNet-50 and ResNet-121. We saw that Residual models are not very helpful when it comes to estimating depth, as both ResNet-50 and ResNet-101 under performed. Also for reducing the network complexity and computational power we also trained the model on MobileNetV2 [48]. Surprisingly, MobileNetV2 performed much much better when compared with other models. If we see the number of parameters and model complexity then ResNet should have performed better than MobileNetV2. But it somewhere shows that it is not just about complex models and deep networks, where we can leave this further for future research.

If we compare our trained best models with the state of the methods that have already been published we can still compare our models with them. We can see those results in Table 2. Our model outperforms the current state-of-the-art despite using fewer parameters (42.6M vs 110M), fewer training iterations (1M vs 3M), and fewer input training samples (50K vs 120K samples). Also if we only compare all the state of the art models that have fewer parameters then I think our Model MobilNetV2 performed exceptionally well, having not so complex architecture.

VI. LIMITATIONS AND FUTURE WORK

The present study [7] has show that depth can be estimated from RGB images using dense networks where they used DenseNet-169 and acheived state of the art performace when they were compared to other methods. One scope of future work in that reseach was to try on more compact networks and different architectures as encoders and record the performance. So in this paper we proposed 4 different methods other than DenseNet-169 which are DenseNet-161, ResNet-50, ResNet-101 and MobileNetV2. When training these architectures due to deep networks they require a lot of computational power and so one limitation was due to lack of computational resources these networks were only trained for 8 epochs and with 7

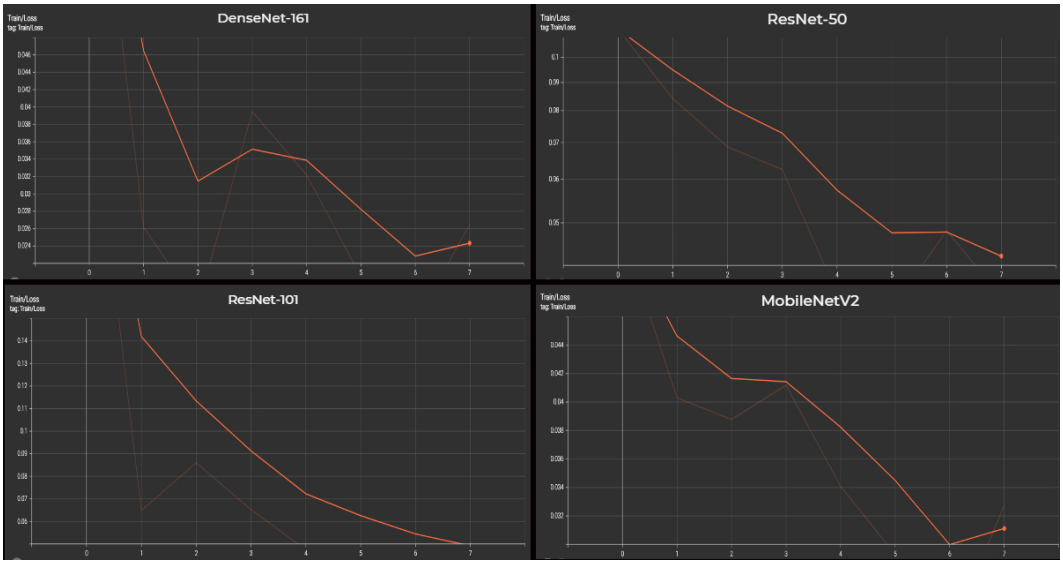

Fig. 8. These are loss graphs obtained from training our models on NYU Depth V2 dataset for 8 epochs.

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	rel $\downarrow$	rms $\downarrow$	$log_{10} \downarrow$
Densenet-161	0.827	**0.969**	**0.993**	0.133	**0.483**	0.056
Eigen et al.	0.769	0.950	0.988	0.158	0.641	-
MS-CRF	0.811	0.954	0.987	0.121	0.586	0.052
Hao et al.	0.841	0.966	0.991	0.127	0.555	0.053
Fue et al.	0.828	0.965	0.992	0.115	0.509	0.051

TABLE II
COMPARISONS OF DIFFERENT METHODS ON THE NYU DEPTH V2 DATASET. THE PROVIDED FIGURES ARE TAKEN FROM THE ORIGINAL PAPERS PUBLISHED.

as a batch size. So one prospect for future scope will be to increase the batch size and increase the number of performance where we can hope to get real images as same as ground truth images. Related to this we also saw that, always using deep networks doesn't necessarily gives you great output everytime as we used MobileNetV2 which is an architecture with less parameter and uses less computatonal power. So other aspect of future work is to study the deep networks and see which aspects are essential for getting real depth images. In this experiment we used a simple decoder which consist of upsampling layers which comparatively gave a good performance. In future, we could also try and use different decoders with different length and layers. As Computer Vision problems are becoming a part of real world problem, finding solution to these problems have became equally important. So Frameworks like Pytorch and TensorFlow, they cam up with new packages which are Pytorch3D and TensorFlow3D respectively. These libraries make it easy when it comes to visualize our images, access datasets, accesss architectures and so on. So in future incorporating these packages in our evaluation can help us model our obtained images as 3D images. It will also help us to visualize the datasets and give access to architectures like Unet architecture.

VII. CONCLUSIONS

In this work, we proposed few convolutional nueural network for depth map estimation on RGB images. We did this by taking use of current advancements in network architecture, as well as the availability of high-performance pre-trained models. We show that a well-built encoder with meaningful weights can have comparable performance to state of the art approaches that rely on either expensive multistage depth estimation networks or the construction and combination of numerous feature encoding layers. We did all the evaluation on NYU Depth V2 dataset. Our goal with this project is to push for higher-quality depth maps that capture object boundaries more accurately, and we've demonstrated that this is doable with existing architectures. We believe there are many more scenarios where normal encoder-decoder models can be used in conjunction with transfer learning to provide high-quality depth estimates.

REFERENCES

[1] W. Lee, N. Park, and W. Woo, "Depth-assisted real-time 3d object detection for augmented reality," in *ICAT*, vol. 11, no. 2, 2011, pp. 126–132.

[2] F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar, "Active refocusing of images and videos," *ACM Transactions On Graphics (TOG)*, vol. 26, no. 3, pp. 67–es, 2007.

[3] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Asian conference on computer vision*. Springer, 2016, pp. 213–228.

[4] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[5] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2002–2011.

[6] A. Saxena, S. H. Chung, A. Y. Ng *et al.*, "Learning depth from single monocular images," in *NIPS*, vol. 18, 2005, pp. 1–8.

[7] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv preprint arXiv:1812.11941*, 2018.

[8] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *arXiv preprint arXiv:1406.2283*, 2014.

[9] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *Advances in neural information processing systems*, vol. 29, pp. 730–738, 2016.

[10] A. Chakrabarti, J. Shao, and G. Shakhnarovich, "Depth from a single image by harmonizing overcomplete local network predictions," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2658–2666, 2016.

[11] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1119–1127.

[12] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5162–5170.

[13] N. Kong and M. J. Black, "Intrinsic depth: Improving depth transfer with intrinsic images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3514–3522.

[14] H. Ha, S. Im, J. Park, H.-G. Jeon, and I. S. Kweon, "High-quality depth from uncalibrated small motion clip," in *Proceedings of the IEEE conference on computer vision and pattern Recognition*, 2016, pp. 5413–5421.

[15] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe, "Multi-scale continuous crfs as sequential deep networks for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5354–5362.

[16] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1, pp. 7–42, 2002.

[17] D. A. Forsyth, "J. ponce computer vision–a modern approach," 2002.

[18] J. Flynn, K. Snavely, I. Neulander, and J. Philbin, "Deepstereo: learning to predict new views from real world imagery," Mar. 13 2018, uS Patent 9,916,679.

[19] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2008.

[20] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 1253–1260.

[21] Y. Horry, K.-I. Anjyo, and K. Arai, "Tour into the picture: using a spidery mesh interface to make animation from a single image," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 225–232.

[22] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 577–584.

[23] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2418–2428.

[24] K. Karsch, C. Liu, and S. B. Kang, "Depth extraction from video using non-parametric sampling," in *European conference on computer vision*. Springer, 2012, pp. 775–788.

[25] M. Liu, M. Salzmann, and X. He, "Discrete-continuous depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 716–723.

[26] M. H. Baig and L. Torresani, "Coupled depth learning," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–10.

[27] J. Konrad, M. Wang, P. Ishwar, C. Wu, and D. Mukherjee, "Learning-based, automatic 2d-to-3d image and video conversion," *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3485–3496, 2013.

[28] S. Choi, D. Min, B. Ham, Y. Kim, C. Oh, and K. Sohn, "Depth analogy: Data-driven approach for single image depth estimation using gradient samples," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5953–5966, 2015.

[29] X. Li, H. Qin, Y. Wang, Y. Zhang, and Q. Dai, "Dept: depth estimation by parameter transfer for single still images," in *Asian Conference on Computer Vision*. Springer, 2014, pp. 45–58.

[30] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 89–96.

[31] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, 2007.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[33] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning–whole book," *Nature*, vol. 521, no. 7553, p. 800, 2016.

[34] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3712–3722.

[35] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3642–3649.

[36] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: a survey," *Optical Engineering*, vol. 58, no. 4, p. 040901, 2019.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] M. Lin, Q. Chen, and S. Yan, "Network in network. corr abs/1312.4400 (2013)," *arXiv preprint arXiv:1312.4400*, 2013.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[40] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[42] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[44] X. Zhang, Z. Li, C. Change Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[46] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2752–2761.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[48] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[49] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[51] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[52] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[54] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[55] R. Ward, X. Wu, and L. Bottou, "Adagrad stepsizes: Sharp convergence over nonconvex landscapes," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6677–6686.

[56] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European conference on computer vision*. Springer, 2012, pp. 746–760.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[58] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

APPENDIX A
IMAGE PREDICTIONS FROM TRAINED MODELS

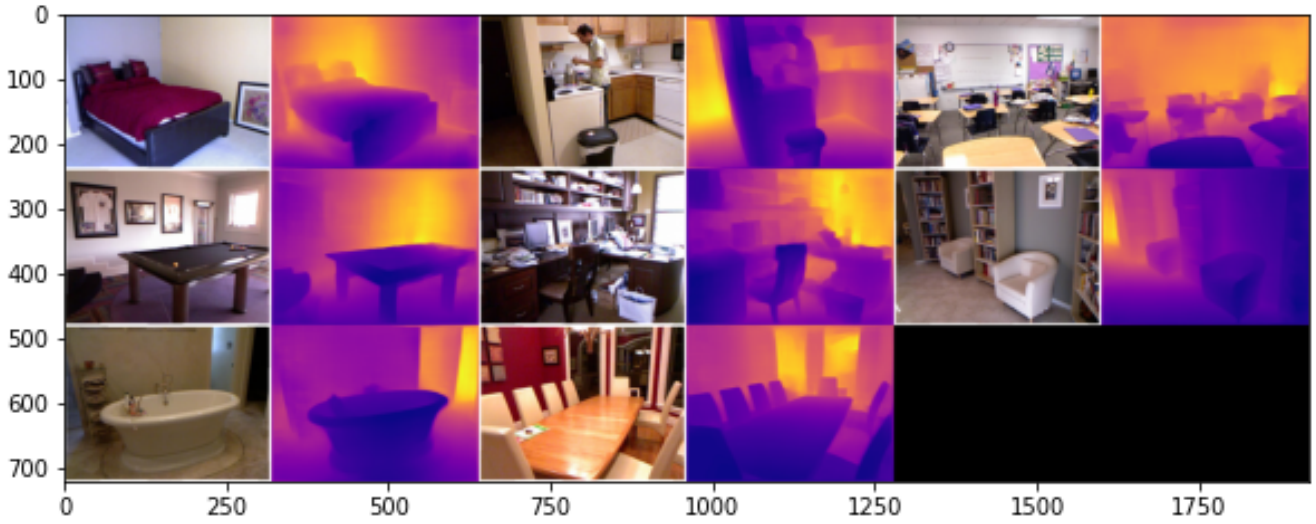

Fig. 9. predictions from DenseNet169

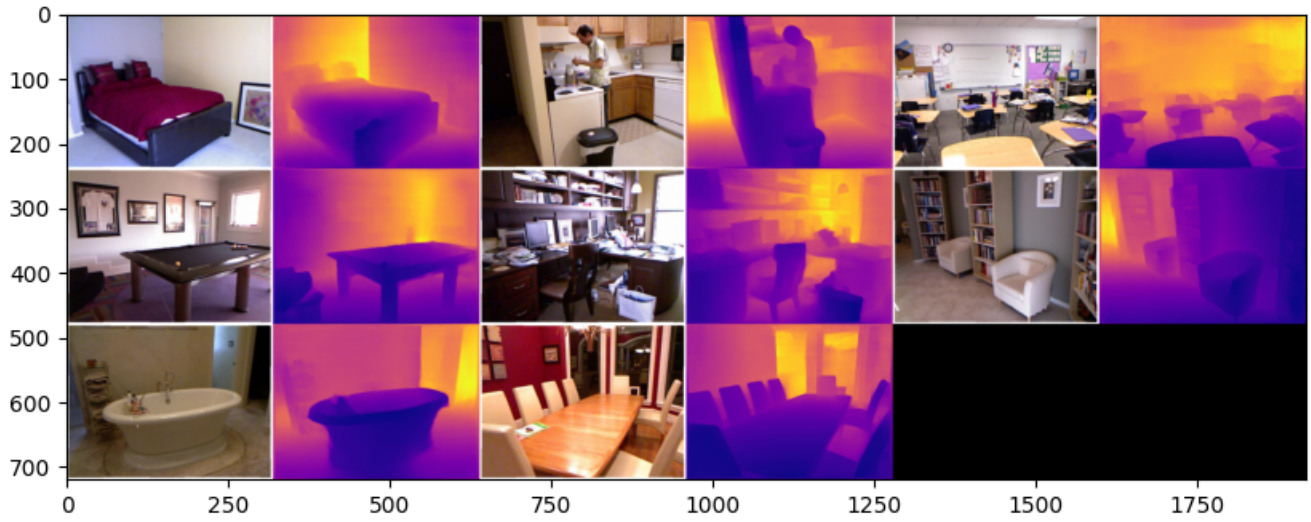

Fig. 10. predictions from DenseNet161

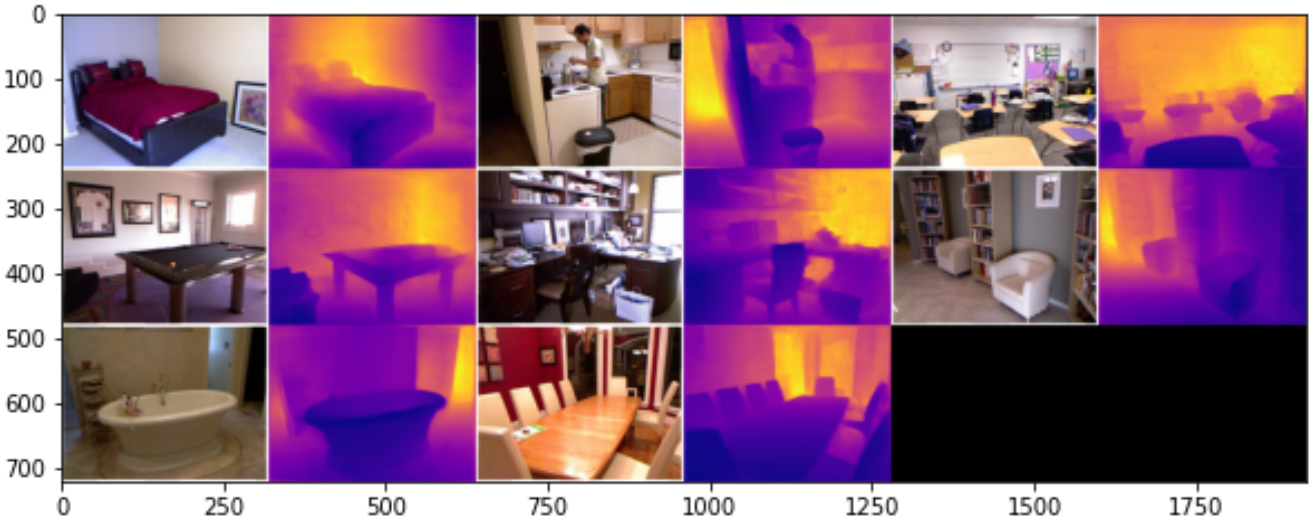

Fig. 11. predictions from MobileNetV2

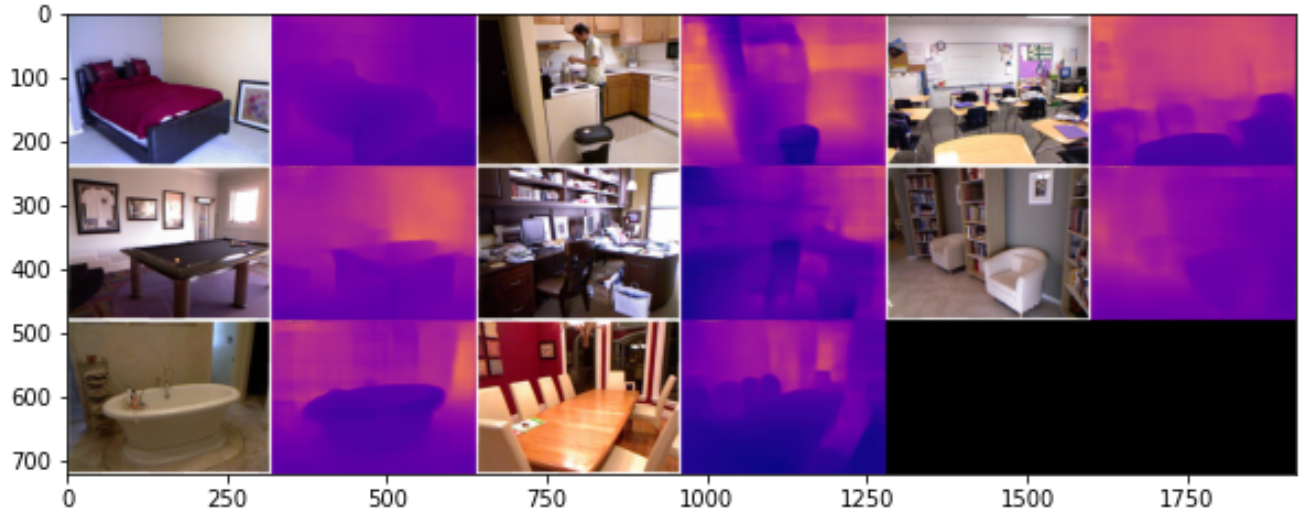

Fig. 12. predictions from ResNet-50

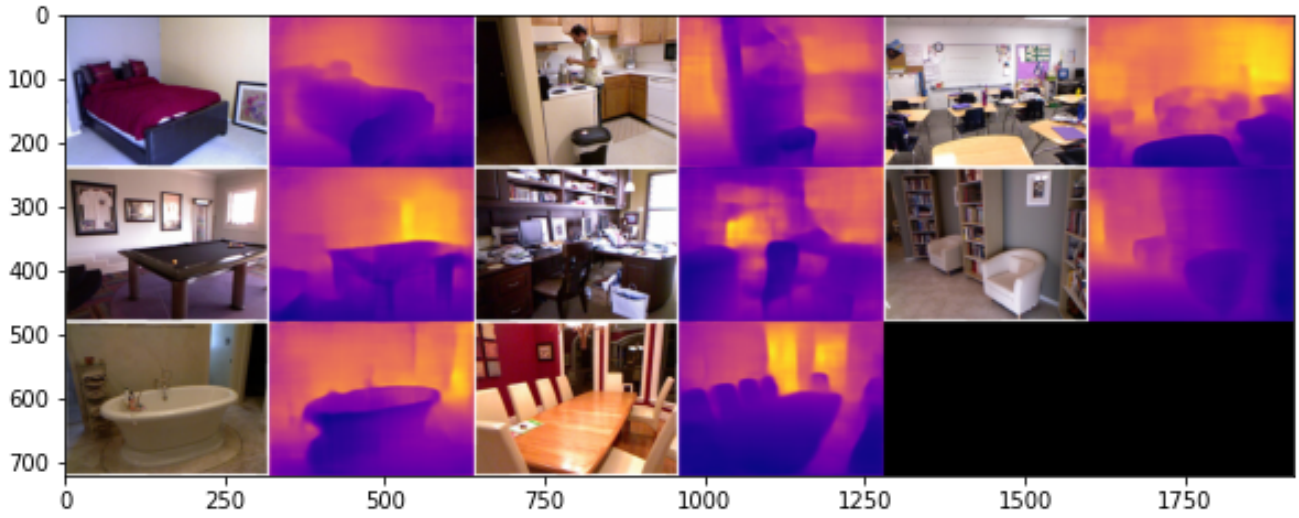

Fig. 13. predictions from ResNet-101