Self-supervised learning: ML Interviews series

Communications between K and A

March 25, 2023

Abstract

We aim to answer questions surrounding the paradigm of learning with limited data (labeled data) while hovering over the basic concepts of machine learning in this regard. The article aims to discuss ML as a communication between two curious novices in the field who want to enjoy learning the nitty-gritty of things in layman's terms. The present article discusses Self-supervised learning (SSL) , SSL is a type of machine learning that allows an algorithm to learn from unlabeled data, making it a powerful tool for tasks such as image recognition, natural language processing, and autonomous driving. Let's see what it has to offer to us.

[code] : github

1 Introduction

The big question, why are even discussing self-supervised learning(SSL)? It comes in the light of a popular ML interview question which is as follows: As we know sufficient and curated data-sets are the backbone of any ML model, and the model is as good as the data. More often than not in real world cases we have abundance of data but not enough labels about them. **Q)** But what's the problem with getting labelled data-set? **Answer:** Labelled data-set often requires human interventions in order to label the data. These often are a part of experimental procedure where a lot of investment is required. e.g. Experiments performed at sophisticated labs to know which molecule (drug) attacks which protein in a cancer cell. Or else taking 1 Million images of dogs and labelling their breed. This procedure of labelling the unlabeled data requires effort and cost at a very high scale. This forms the motivation to find a technique which can better the performance of supervised learning with limited amount of data. The **objective** of this paragraph was to establish the motivation behind learning self-supervision. Next up, let's see some motivating examples of SSL

The idea of self-supervision is taking the ML world by storm. To show its power I would suggest that we all have used Chat GPT to simplify our lives. The 'PT' in 'GPT' stands for pre-training which is another name for self-supervised learning. It might be a bit confusing at first, because we've just seen examples of self-supervision in the case of images. But the idea of self-supervision can be applied to various cases, the most popular being Image and CV related models, Language models etc. **Q)** What makes Chat GPT a self-supervised model and not a supervised model? **Answer:** ChatGPT is not a supervised learning model because it is trained using a self-supervised learning approach called language modeling. In supervised learning, a model is trained on labeled data where each input is paired with a corresponding output label. In contrast, self-supervised learning is a type of machine learning where a model is trained on unlabeled data, and the model generates its own labels based on the structure of the data. In the case of ChatGPT, the model is trained on a large corpus of text data without any explicit labels. During training, the model learns to predict the next word in a sequence of text based on the context provided by the previous words. This is a self-supervised learning task because the model generates its own labels based on the structure of the language, rather than being given explicit label. Whereas, A super-vised learning task in similar setting would be something like, Given a sentence predict the sentiment of this sentence; henceforth the training dataset would have a corpus of sentences and labelled emotions or sentiments. It is a very popular NLP task, it works on extracting the features of the sentence and mapping it with the labels during training. The features would be frequencies of certain words, semantic relation between words, occurrence of specific parts of speech etc. The **objective** of this paragraph was to establish the case of self-supervision and show

a detailed example of how self-supervision is different from normal supervision by taking the case of one of most popular works in this field.

Since, we are talking about supervised learning and unsupervised learning, it is also worth noting, that given a dataset; the learning task (supervised) can differ greatly depending on the provided label. For example, Given a picture of an animal we can have various labels like name of the animal, breed of the animal, sex of the animal, age of the animal, number of teeths, facial expression of the animal and the list goes on. The learning outcome of each of these labels will be different. As from the list, our learning may answer the following question in order; Is it a Dog or a Cat or a lazy Sloth? (The label being 'name of the animal') Is it a Maltese or a Shitzu? (The label being 'breed of the animal') Is it a male giraffe or a female one? How old is this animal? How many teeths does this animal have? and so on. Henceforth, a decent ML model will be able to answer these questions respectively on a brand new image (test image). The **objective** of this paragraph was to give a side-flick on understanding supervised task and its relation with the dataset. Next paragraph, we return back to self-supervision in the case of Computer vision which we will stick to, for the most part of our further discussion.

But our current task is to deal with the question of 'using abundant unlabeled data and limited labeled data (data of the same kind, i.e. labeled data of dog breed images and unlabeled data of MNIST number images do not help !). Can we use abundant unlabeled data to improve our supervised work? i.e given 10 samples of images of dogs with their breed labels; can we really predict the breed of an new and unseen image of dog? With only 10 labelled images, the supervision is likely to fail (**Q** How are you so confident that it will fail? Is it just based on experience? **Answer:** Well yes, based on experience we know it will fail; but the maths of the problem can also be used to answer this question. Given that we have a clear image we would be having a minimum of 64x64 pixel with 1 channel; that would a minimum of 5 down-stepping convolution layers which may minimum of have 30 parameters each; thus 150 parameters only to vectorise the image. Finding the optimum value of 150 parameters with 10 samples seems to be a difficult task for even the best optimiser in the world!). The **objective** of this paragraph was to re-introduce the problem which we will be looking at in the results.

2 Some of the book-shelf examples of SSL

Let's see some cases where self-supervised learning is being used these days.

- **Image Classification:** A common technique in self-supervised learning is to use a pretext task. For example, a model may be trained to predict the rotation of an image, or predict which patch of an image is missing. By solving these tasks, the model learns useful representations that can be used for other tasks, such as image classification or object detection.[CKNH20, GSA+20, HFW+19, CKS+]
- **Language Modeling:** Another example of self-supervised learning is language modeling. A model is trained to predict the next word in a sequence of text. By learning the underlying structure of language, the model can generate coherent sentences, and can be fine-tuned for other natural language processing tasks, such as sentiment analysis or machine translation. [DL15, RNSS18, LOG+19, YDY+19, DCLT19]
- **Audio Processing:** A model can be trained to predict a small portion of an audio clip, given the rest of the clip as input. By solving this task, the model can learn to recognize different sounds and distinguish between them, such as speech, music, or background noise. [NLK21, CF21, KJKL21, WKL20]

These are just a few examples of self-supervised learning. The key idea is to leverage the structure of the data itself to train a model, without the need for explicit labels or annotations.

2.1 How exactly SSL helps?

We have now seen plenty of examples of how self supervision helps us by pre-training our model such that it performs better on the downstream supervised task (with limited labeled data). It is now time

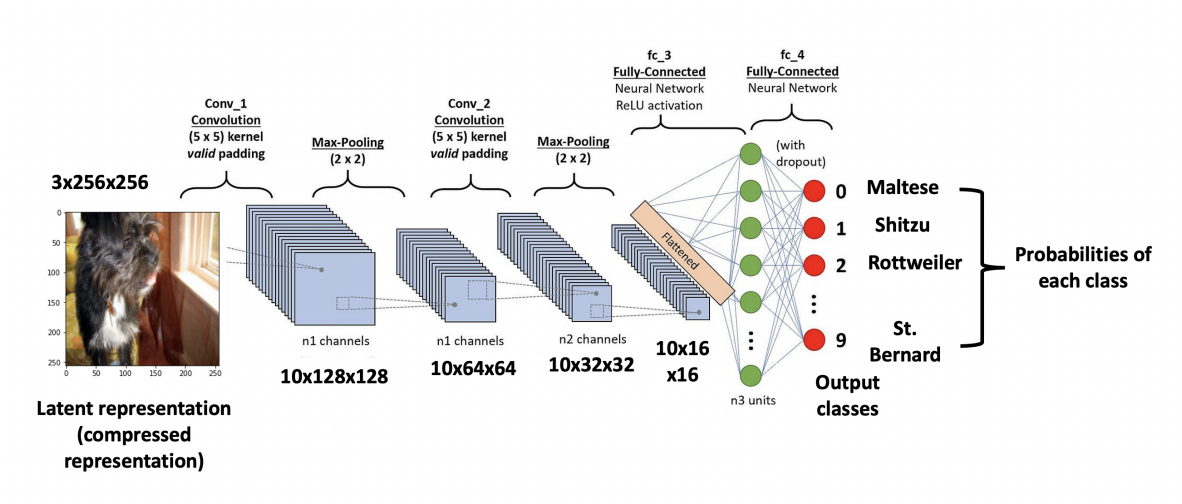

Figure 1: Training pipeline of CNN with images.

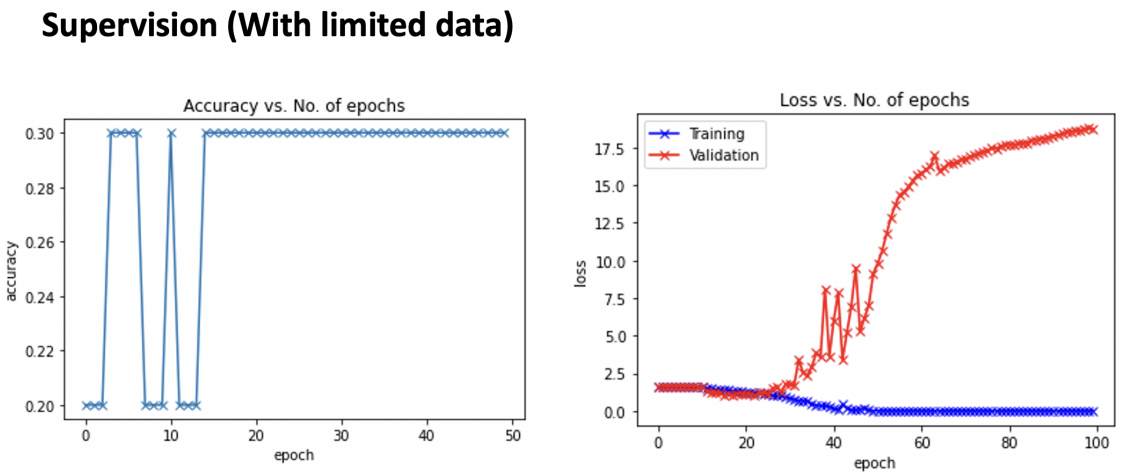

Figure 2: Accuracy and loss plots of training of CNN with images.

to de-mystify what pre-training actually does and how it helps? For this we will take an example of supervised learning task of predicting breeds of dogs from their images. We will develop an end-to-end learning framework for this example and try to show the effects of pre-training on limited supervision. Our dataset will consist of images of dogs classified according to their breeds, however we will only use a few classes with a few images in each class for our supervised learning work. After achieving a certain accuracy with this limited supervision, we'll set up our pre-training problem and try to get some insight into what pre-training will do to our downstream results.

First come first, let's train our CNN model on the dataset, Figure 1 shows the model architecture which seems pretty straightforward. The model involves a few layers of (Convolution,pooling) pair. Moving forward we flatten the final vector in order to feed the input to a feed-forward NN. The final outputs to which maps to the 10 breeds (classes) we wish to classify our images into. The final output will be in terms of soft-max probabilities with usual (0-1) probabilistic range of measure.

The training proceeds as usual with decreasing training loss and increasing model accuracy (computed on validation set) as seen in Figure 2 But soon enough we observe increasing validation loss and decreasing training loss, which is a clear indication of model over-fitting. The model has learned all the samples (as the number of samples are low), rather than learning the features of the training samples. Hence, it is not able to generalise at all on the unseen dataset.

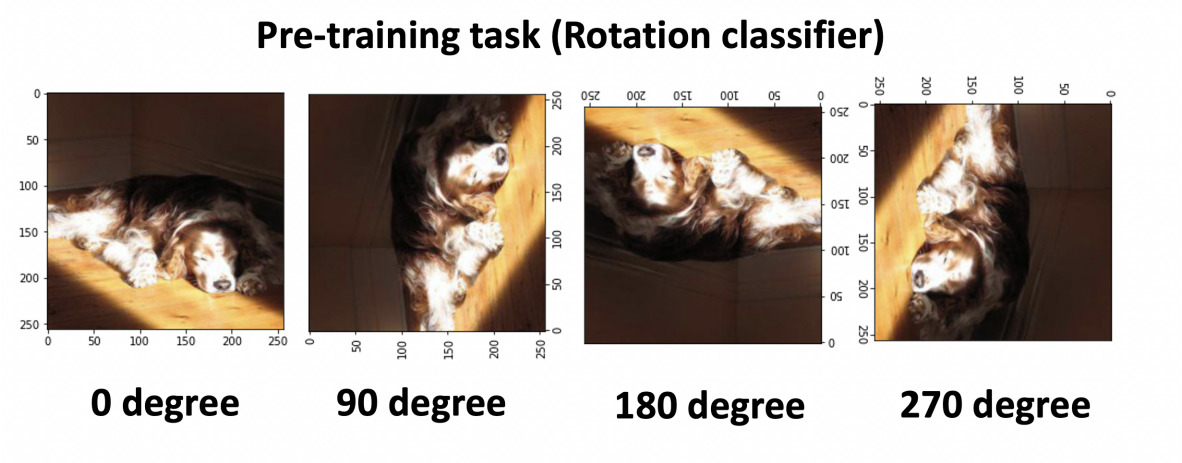

Figure 3: Construction of a labelled dataset for our pre-training task.

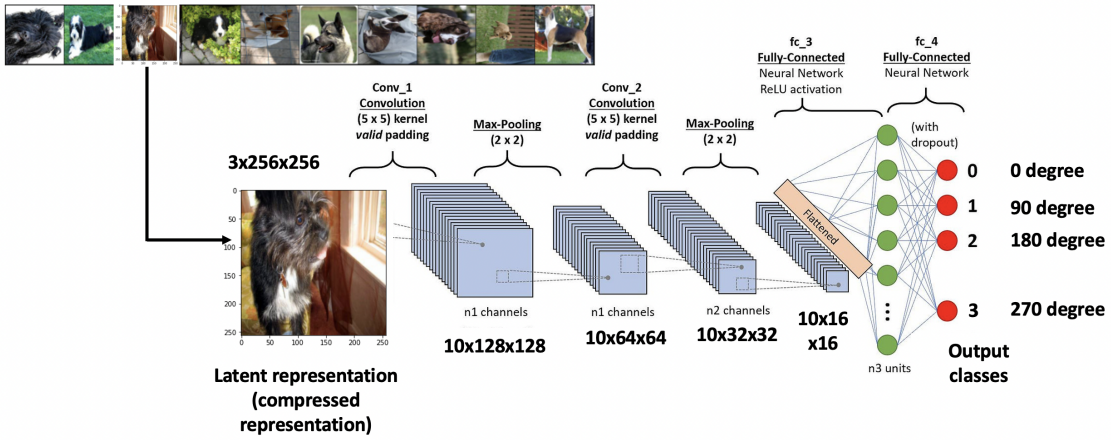

Figure 4: Pipeline of the pre-training task.

The results are pretty bad as expected, but can we improve it somehow? Yeah, let's go out to a dog park (or a few dog parks) and click several pictures of these wonder-full creatures. It's not necessary to know the breeds, we just need the pictures nothing else! That would not be a difficult task, right? With all these unlabelled images we will construct our own pre-training paradigm such that we can learn some useful representations from these bunch of images. We can have a lot of pre-training task with this unlabelled data, and we choose the pre-training task of rotation predictor. That is, simply take an image and rotate it by $[0, 90, 180, 270]$ degrees and form a labelled dataset of your own as shown in Figure 3.

After we have created our own labelled dataset of rotated images, i.e from 1 unlabelled image we created 4 labelled images, as shown in Figure 3. The next task is to define a rotation predictor model, i.e the model will predict the angle of rotation on unseen images. Let's look at the model pipeline in Figure 4. The model looks pretty much the same as before, just change the output heads. Now let's see the results of this training.

The pre-training results look great! Given that the model is trained on 40k images and validated of 5k images, the results seem to be well generalised. It is important for this model to be generalisable, as we require the network to learn the features of the images rather than learning the whole image-label pair. We choose a geometric pre-task, such that the model probably learns how a dog looks like, how

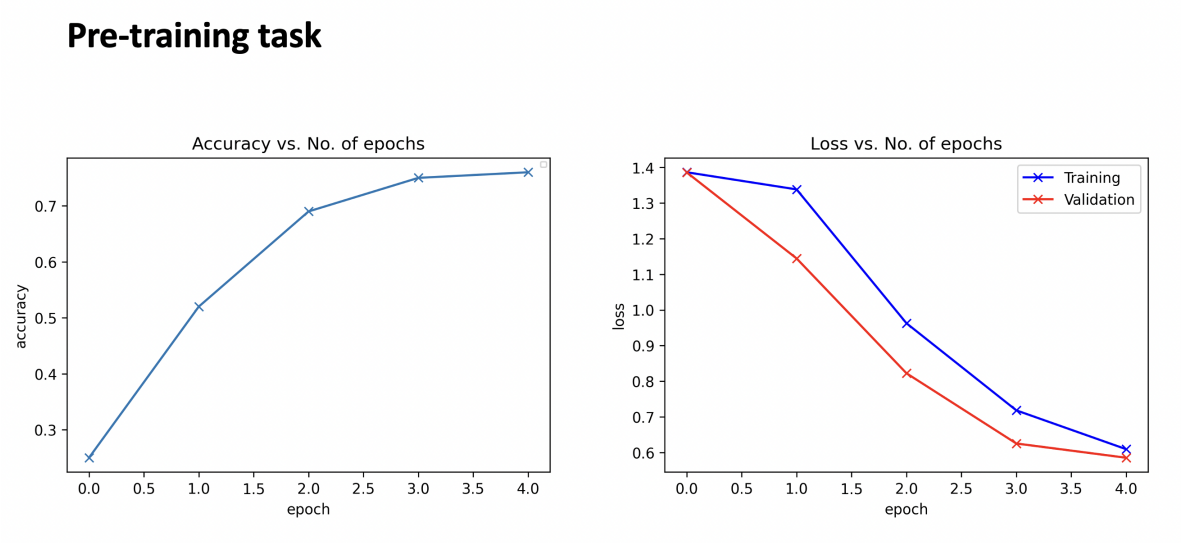

Figure 5: Accuracy and loss plots of the pre-training task.

they have fluffy hairs, curved tails, four legs, etc. Moreover, you can always increase the number of kernels in each conv layer if you wish to learn more features. But usually for images of animals, 10 features are enough (that's what we have used in our model).

All things working as planned up till now, let's move forward to the more exciting part; which is the downstream task. We will now do the same task of supervised learning of dog breeds as we did before. But we will now use the same learned weights, which we have gotten from the self-supervision (pre-training task) as shown in Figure 6. An important observation here is that, we will only use the weights of conv layers and not the feed-forward parts. This is important as the conv layers only have the learned feature representations and not the feed-forward layer. Hence, we choose to do so.

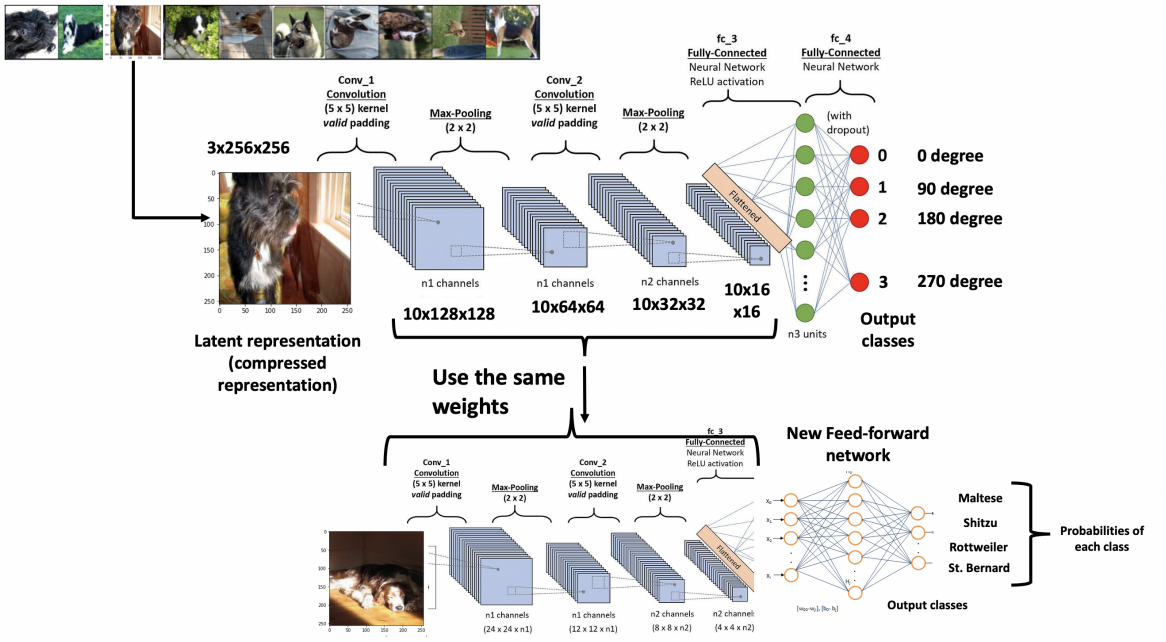

Figure 6: Training of CNN with SSL (Pre-training) with images.

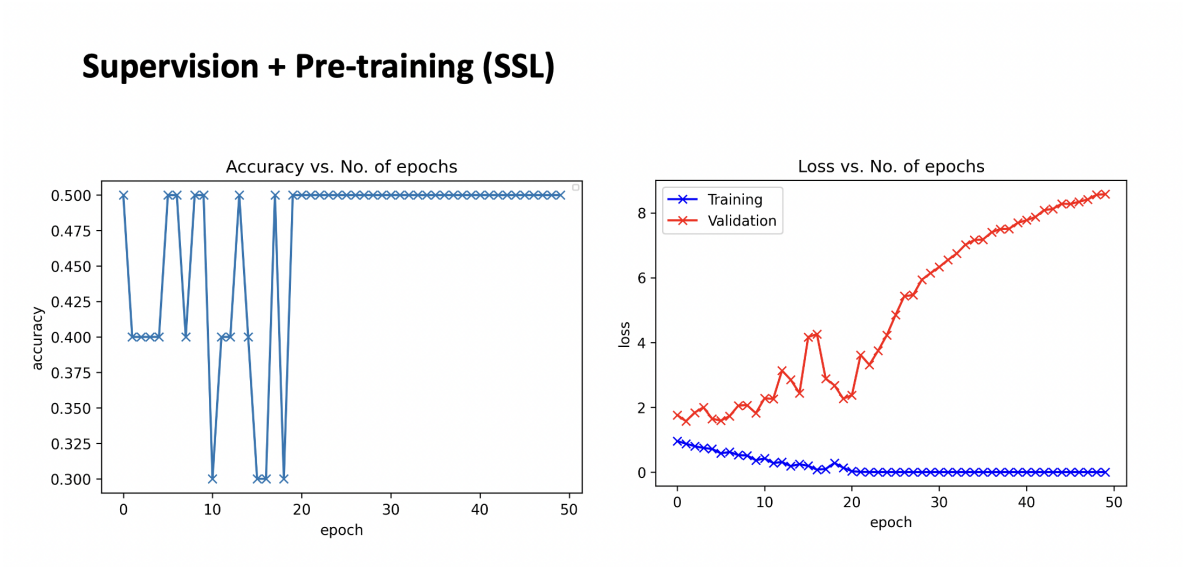

Figure 7: Accuracy and loss plots after pre-training.

The results of this model is significantly better than its Figure 7 , also lower validation error accumulation. Although it is worth noting that the model is still over-fitting, which is an evident pain point and can't be neglected. This may be happening due to two possible reasons ; firstly, the feed-forward network itself maybe memorising the images (as there are really very few total number of images = 567) so we can definitely decrease its complexity and try again. Secondly, maybe the limited dataset has enough feature complexity to have this level of complexity in the network but the dataset is so small that we will always get this kind of behaviour, to get over this issue we would need to have checkpoints for finding the best model while our trainer is running!

Let's look at an example test case of prediction with this supervised model. Our example case, is of a crazy being named 'Shiro' Figure 8 (His picture has been taken without consent. Apologies; if he happens to come across this article anytime in the future.) We clearly see the model was now able to correctly classify his breed! Which really seems to be a good enough justification (at-least for him) for this method to be gaining popularity in recent times.

2.2 More complex SSL task in Computer vision : Auto-pilot

Self-supervised learning has also been applied in the domain of autopilot systems for autonomous vehicles[ZXW+, GSS20, WSCL18, PAE+17, BDTD+16] and computer vision[DGE15, GK18, CKNH20, GSA+20, CWG21]. Here are some examples of self-supervised learning tasks used in autopilot systems:

- **Depth estimation:** Depth estimation involves predicting the distance of objects from the camera. A self-supervised approach to this task involves training a neural network to predict the relative depth of pixels in a monocular image by comparing it with another image taken from a different viewpoint.
- **Optical flow prediction:** Optical flow prediction involves predicting the motion of objects in a sequence of images. A self-supervised approach to this task involves training a neural network to predict the optical flow between two consecutive frames in a video sequence.
- **Scene reconstruction:** Scene reconstruction involves generating a 3D representation of the environment from 2D camera images. A self-supervised approach to this task involves training a neural network to predict the camera pose and 3D structure of the environment by minimizing the reprojection error of the 2D image points onto the reconstructed 3D scene.

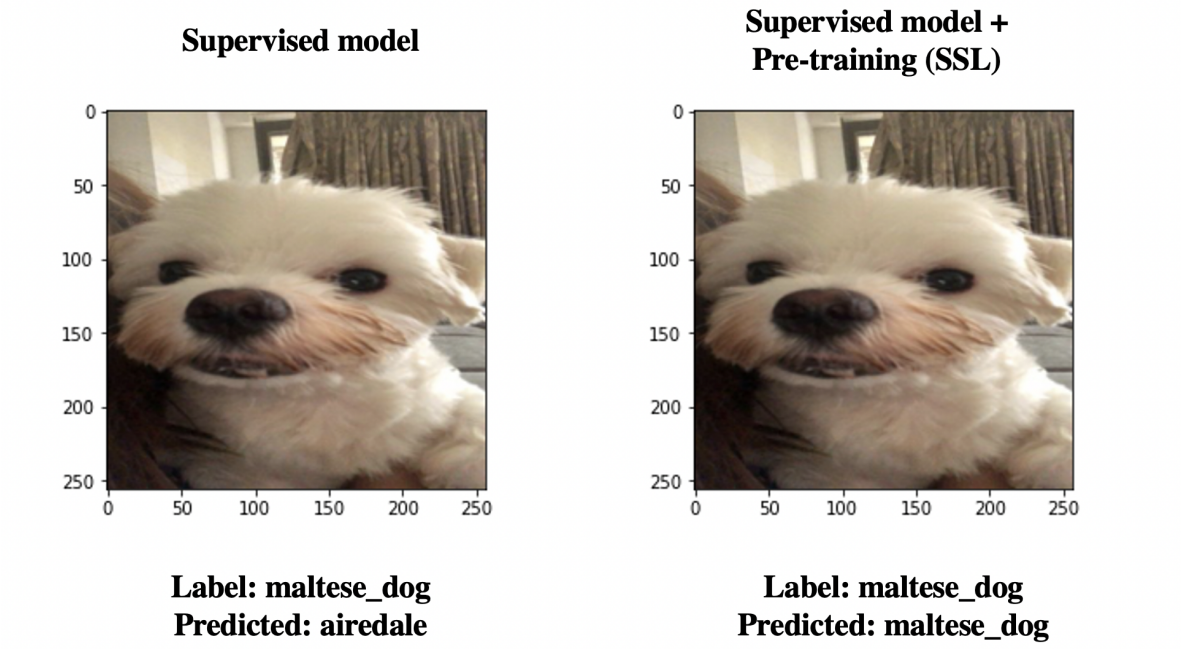

Figure 8: Prediction on specific test case.

- **Action prediction:** Action prediction involves predicting the intended actions of other vehicles and pedestrians in the environment. A self-supervised approach to this task involves training a neural network to predict the motion of objects in the scene and their interactions with the ego vehicle.

These self-supervised tasks can enable an autopilot system to learn useful representations of the environment and the behavior of other agents without the need for large amounts of labeled data. This can lead to more robust and adaptive autopilot systems that can operate in a wide range of driving scenarios.

2.3 SSL and realted concept Transfer learning

Finally, we need to make a comment about a related concept called Transfer learning, as our model architecture uses transfer learning as part of our SSL procedure. Transfer learning and self-supervised learning are related concepts, but they are not exactly the same thing. Transfer learning is a method where a pre-trained model is used as a starting point for a new task, rather than training the model from scratch on the new task. The pre-trained model may have been trained on a large labeled dataset, such as ImageNet, using supervised learning or some other method. The idea is that the pre-trained model has already learned useful features that can be transferred to the new task, which may have a smaller labeled dataset.

Self-supervised learning, on the other hand, is a type of training where the model learns to predict a label or target from the input data itself, without using any explicit labels from a labeled dataset. In other words, the model is trained to learn useful representations or features from the data, which can then be used for other tasks. Self-supervised learning can be used as a form of pre-training for transfer learning. The idea is to train a model using a self-supervised task, such as predicting the missing part of an image, and then transfer the learned features to a new task, such as image classification. So, while transfer learning and self-supervised learning are related concepts, they are not the same thing. Transfer learning involves using a pre-trained model for a new task, while self-supervised learning

involves training a model to learn useful representations or features from data without using explicit labels. Self-supervised learning can be used as a pre-training method for transfer learning, but it is not the only method.

References

[BDTD+16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[CF21] Paul Cramer and Tim Fingscheidt. Self-supervised learning for audio event detection using contrastive predictive coding. In *International Conference on Machine Learning*, pages 2026–2036. PMLR, 2021.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Advances in Neural Information Processing Systems*, 2020.

[CKS+] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Contrastive multiview coding.

[CWG21] Xiangyu Chen, Tongtong Wang, and Boqing Gong. Exploring simple siamese representation learning for unsupervised instance segmentation. In *Proceedings of the International Conference on Machine Learning*, pages 1728–1738, 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[DL15] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, pages 3079–3087, 2015.

[GK18] Spyros Gidaris and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proceedings of the International Conference on Learning Representations*, 2018.

[GSA+20] Jean-Bastien Grill, Florian Strub, Florent Altche, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, R'emi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *International Conference on Learning Representations*, 2020.

[GSS20] Ankit Garg, Julian Serrano, and Ashutosh Saxena. Learning to drive using inverse reinforcement learning and deep q-networks. *IEEE Robotics and Automation Letters*, 5(2):3813–3820, 2020.

[HFW+19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Advances in Neural Information Processing Systems*, 2019.

[KJKL21] Minseok Kwon, Ju-ho Jeong, Taesup Kim, and Hoirin Lee. Self-supervised learning for audio-visual speaker diarization. In *International Conference on Machine Learning*, pages 5669–5679. PMLR, 2021.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[NLK21] Dat T Nguyen, Youngmoon Lee, and Hanseok Ko. Improving speech emotion recognition via self-supervised learning. In *International Conference on Machine Learning*, pages 7552–7562. PMLR, 2021.

[PAE+17] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, Trevor Darrell, and Jitendra Malik. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. In *International Conference on Learning Representations*, 2017.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[WKL20] Myungjong Won, Woojay Kim, and Kyogu Lee. Self-supervised learning for speech separation by exploiting global information. In *International Conference on Machine Learning*, pages 10015–10025. PMLR, 2020.

[WSCL18] Yonghong Wu, Mike Schuster, Zhifeng Chen, and Quoc V Le. Unsupervised feature learning and deep learning: A review and new perspectives. In *Proceedings of the IEEE*, volume 106, pages 21–49. IEEE, 2018.

[YDY+19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, pages 5754–5764, 2019.

[ZXW+] Chaoyang Zhang, Fanyi Xiao, Meng Wang, Zijie Liu, and Jianguo Zhang. Learning to drive by learning to predict. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12154–121.