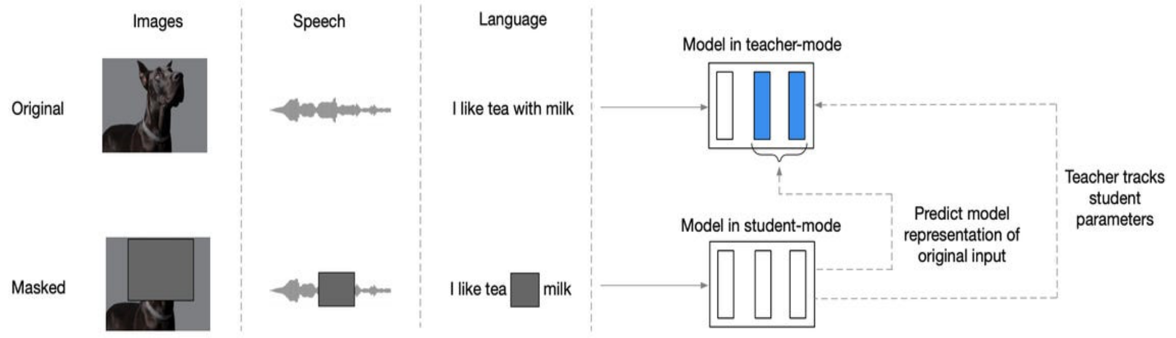
Unified Self-Supervised Algorithm for Speech, Vision, and Text: Meta's data2vec Project

Obafemi Jinadu

Introduction to Machine Learning (COMP 135)



[1]

The data2vec project is primarily the work of Facebook (now Meta) AI researcher Alexei Baevski, data2vec is a self-supervised learning framework that uses the same learning method for either speech, natural language processing (NLP) or computer vision [1]. Self-supervised learning builds representations of data without human-annotated data, conventional self-supervised learning today largely focuses on a single modality, building tailored models to work solely on the individual modality. NLP models are trained to fill in blanks or detect sentiments from text data and predict texts. Speech models, learn a catalog of basic sounds to predict missing sounds. In computer vision, models are trained on image data to predict images or pixels within a class. The data2vec is a unified framework for self-supervised learning in three modalities: speech, texts, and images. However, it still learns representation separately for each of the three modalities. The image above captures the workings of data2vec, which is a single model operating in two modes, the teacher mode, and the student mode. In the *teacher mode*, sample data (speech, text, or image) is embedded into tokens termed representations, this serves as the training target. In the *student mode* [5], a

[1] Ray Tiernan, An overview of Meta's data2vec approach, online image, ZDNet, January 20, 2022, https://www.zdnet.com/article/metas-data2vec-is-the-next-step-toward-one-neural-network-to-rule-them-all/.

masked version of the same sample is passed where the student predicts the model representation of the full unmasked sample data (i.e., the training target). The target representations encode all the information in the training sample and the learning task is for the student to make predictions on these representations given a masked view of the input i.e., reconstruct these latent representations, learning is achieved by minimizing the objective function (Smooth L1-Loss) between the prediction the student makes and the training targets. The student network tries to develop its sense of the data by reconstructing what the teacher network has already built (training target).

Most self-supervised learning techniques involve getting a model to take some input data (e.g., image, text, or speech) and mask out certain components of the input data (e.g., blacking out pixels or words) to get the model to predict the masked components. Masking these inputs makes the task hard enough to force models to learn facts about the data that enable optimized generalization. This approach has an inherent limitation as the task of filling in the blanks differs across modalities. To solve this, data2vec is trained not to fill in blanks like conventional semi-supervised learning models. Instead, data2vec uses *contextualized latent representations* of images, text, and speech generated by a teacher network, this provides a common task that can be used regardless of the modality/input data type. However, to obtain these common latent representations, different encodings are applied per modality.

The model leverages a standard Transformer neural network architecture [11] where the teacher and student are transformer networks, for each of the three modalities, a unique encoding strategy to the input data is applied. The transformer neural network architecture was used because of its ability to extract contextual representations. The transformer network takes in encoded representations of the sample input, these encodings map features of the sample in a space where similar features are closer to each other. These encodings could be pre-trained. An example of this in NLP is the GloVe embedding (provided in project 2). A positional encoder is then applied to the encoded input, this is a vector that gives context to features based on the position of the feature in a sample, the output of applying the positional encoder to the input embeddings are vectors that have positional information (context). This vector goes into the encoder block of the transformer network, the encoder block consists of a multi-head attention layer and a feed-forward neural network layer. The attention layer helps capture the parts of the input that should be focused on, and it does this by generating attention vectors that show how relevant a feature is relative to other neighboring features (it captures contextual relationships between features in a sample). Attention vectors are then passed to a feed-forward network, the feed-forward networks are used to transform the attention vectors into forms that are digestible by the next block in the abstraction which is a linear layer (which is another feed-forward network) and it is used to expand the dimensions into the number of classes in the model. Finally, the output from the linear layer is passed into a softmax layer which as taught in class is a normalized logistic function where the probability distribution sums up to 1 and the class with the highest probability value is the prediction. As we learned in class, a standard feed-forward neural network consists of one or multiple layers of "neurons" joined together in a graph structure [10]. The first layer takes in the input (For example, the output of the first feed-forward network which serves as the input for the linear layer - which is the second feed-forward network), this input is fed into the network, a set of numerical features that characterize the sample data. The final layer in the network generates the output, the number of output neurons corresponds to the number of classes. In between, the first and last layer there may be one or multiple hidden layers, each of which consists of neurons that connect surrounding layers. For each connection, the neuron stores a weight. Values are passed into the first layer, and each neuron at the next layer takes all these parameter values and computes the weighted sum over all these values, before applying an often times nonlinear, differentiable activation function g(in$_j$) which could be Identity (here g(in$_j$) = in$_j$), Logistic, ReLU, softplus, Gaussian, hyperbolic tanh, etc., to the weighted sum (g, in this case, is softmax). The result of this process is the value that the neuron passes along to the neurons in the next layer down, which repeats the process all the way to the last layer. The output of this linear (feed-forward network) are vectors of values, one per neuron in the final layer i.e., the final layer consists of the number of all possible classes that can be predicted and the softmax function converts these vectors into a probability distribution where the highest probability value is the selected class. Standard feed-forward networks are typically used for classification tasks, where it begins by taking in training data as input (these are data points with labels – known targets). The network is set to apply random weights on all its connections, training is then achieved by minimizing the output error which is the average difference between what the label predicts and the ground truth (what we know to be the true label). The training of these multi-layer networks is achieved by backpropagation [3, 9], this involves iteratively computing results of the training set by making a forward pass from input to output neurons, the error in each iteration is measured, and the weights are adjusted backward from output neurons to the first hidden layer. The number of iterations can be adjusted as a hyperparameter by the user, this along with several other hyperparameters can be tweaked for optimal performance, some of these hyperparameters are learning rate, hidden layer size, activation function, solver etc.

For the computer vision encoding, 224 x 224-pixel images are embedded into 16 x 16 [2] patches each. Each patch is linearly transformed and a sequence of 196 encoded representations is generated to fed in as the input to the transformer. Blocks of multiple randomly sampled adjacent patches are masked (a block contains at least 16 patches and about 60% of the sequence of encoded representations are masked). For image classification, mean-pooling (pooling as treated in class, is down-sampling to reduce volume size) is applied to the output of the last transformer block over all patches and this serves as input to a softmax layer performing classification. To evaluate, the model was pretrained on ImageNet-1k and a downstream task was created to predict a single label for each image of similar datasets. A pre-trained model is a saved network (weights) that was previously trained on a large dataset i.e., a model created by someone else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, the pre-trained model could be used as a starting point.

For the speech-processing data encoding, the input samples are 16 kHz waveforms, and a 1-D multilayer *convolutional neural network* is applied. Unlike 2-D convolutional neural networks for images (segmented by space) which can have 2 or 3 channels for the case of greyscale and colored images respectively, 1-D convolutional neural networks are used for temporal data like speech (data segmented by time) with scalar channel dimension of size 512. This results in an encoder output frequency of 50 Hz with a stride of 20ms between each sample. For masking, time steps were uniformly sampled, for each sample, the 10 subsequent timesteps are masked (for student prediction, ~49% of all time steps are masked). To evaluate, the model was pretrained on the Libri-light low-resource labeled data, a 960-hour speech audio data, the evaluation metric for speech processing used is called word error rate (WER).

The convolutional neural networks used in encoding speech samples date to work from the late 90s [8]. In these networks, multiple layers of networks are used to extract data from the input by virtue of a kernel of specified size sliding across input data which could be 1-D (temporal input representation), 2-D (spatial input representation), or 3-D input (spatial and temporal input representation e.g., x, y spatial axis and a time, t axis). The process of learning with these complex networks- now popularly known as *deep learning* has been made possible by advances in computing hardware and algorithms with the capacity to train massive [4, 6] datasets.

For text/natural-language-processing data encoding, the input data (a corpus of words) is tokenized by embedding word units in distributional space as embedding vectors and 15% of the tokens are replaced with masks. The model was pre-trained on the Books Corpus [13] and English Wikipedia. The model was evaluated on the General Language Understanding Evaluation (GLUE) benchmark [12]. GLUE is a framework for evaluating and analyzing natural language understanding systems and it consists of a diagnostic dataset designed to evaluate model performance with respect to a vast range of linguistic phenomena found in natural language, it also consists of a public leaderboard for tracking performance on the benchmark.

Data2vec reports state-of-the-art performance on all three modalities (speech, image, and texts) outperforming self-supervised models built to handle single domains such as BERT (Bidirectional Encoder Representations from Transformers) a transformer-based machine learning technique for natural language processing, HuBERT (Hidden unit BERT) a self-supervised speech representation learning for speech recognition, and MaskFeat (Masked Feature Prediction) a self-supervised model for computer vision.

A challenge common to deep networks is interpretability. For example, in a convolutional neural network model built for image classification, the inner workings of these models typically remain unknown, while we suspect that for an image classification task, the first set of layers detects patterns like edges, corners, and deeper layers detect more complex patterns in the image, we usually cannot be certain. This is more so the case in data2vec as it handles data across multiple modalities, this shows there is often a tradeoff between simpler, interpretable models with often poorer performance and more complex models that that often have better performance but little explainability. This speaks to the need for a new generation of interpretability techniques to keep up with scaled multi-modal systems.

In data2vec, latent variables are not a combined encoding of the three modalities. A separate model is trained for each modality but the process through which models learn is identical. Therefore, data2vec is not a universal architecture. This could potentially change as the authors of this work plan on combining data2vec with Google AI's recently published work on an architecture called Perceiver [7], which uses a single preprocessing technique for all input data domains (speech, image, text, etc.). Exciting machine learning applications like this that allow generalized learning across multiple modalities beg a question about the relatability of these three modalities. For example, does data2vec detect a relationship between the latent representation generated for a dog image and a latent representation generated for the word "dog" or a speech sample data that says "dog". This could potentially tell us more about the robustness of the concept the model learns.

References

[1] Baevski, Alexei, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. "Data2vec: A general framework for self-supervised learning in speech, vision and language." *arXiv preprint arXiv:2202.03555* (2022).

[2] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

[3] Dreyfus, Stuart E. "Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure." *Journal of guidance, control, and dynamics* 13, no. 5 (1990): 926-928.

[4] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[5] Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch et al. "Bootstrap your own latent-a new approach to self-supervised learning." *Advances in Neural Information Processing Systems* 33 (2020): 21271-21284.

[6] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18, no. 7 (2006): 1527-1554.

[7] Jaegle, Andrew, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. "Perceiver: General perception with iterative attention." In *International Conference on Machine Learning*, pp. 4651-4664. PMLR, 2021.

[8] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

[9] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323, no. 6088 (1986): 533-536.

[10] Russell, Stuart, and Peter Norvig. "Artificial intelligence: A modern approach. third edit." *Upper Saddle River, New Jersey* 7458 (2010).

[11] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[12] Warstadt, Alex, Amanpreet Singh, and Samuel R. Bowman. "Cola: The corpus of linguistic acceptability (with added annotations)." (2019).

[13] Zhu, Yukun, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books." In *Proceedings of the IEEE international conference on computer vision*, pp. 19-27. 2015.