

Information-Theoretic Probing with Minimum Description Length

Elena Voita^{1,2}

Ivan Titov^{1,2}

¹University of Edinburgh, Scotland

²University of Amsterdam, Netherlands

lena-voita@hotmail.com ititov@inf.ed.ac.uk

Abstract

To measure how well pretrained representations encode some linguistic property, it is common to use accuracy of a probe, i.e. a classifier trained to predict the property from the representations. Despite widespread adoption of probes, differences in their accuracy fail to adequately reflect differences in representations. For example, they do not substantially favour pretrained representations over randomly initialized ones. Analogously, their accuracy can be similar when probing for genuine linguistic labels and probing for random synthetic tasks. To see reasonable differences in accuracy with respect to these random baselines, previous work had to constrain either the amount of probe training data or its model size. Instead, we propose an alternative to the standard probes, information-theoretic probing with *minimum description length* (MDL). With MDL probing, training a probe to predict labels is recast as teaching it to effectively transmit the data. Therefore, the measure of interest changes from probe accuracy to the description length of labels given representations. In addition to probe quality, the description length evaluates ‘the amount of effort’ needed to achieve the quality. This amount of effort characterizes either (i) size of a probing model, or (ii) the amount of data needed to achieve the high quality. We consider two methods for estimating MDL which can be easily implemented on top of the standard probing pipelines: variational coding and online coding. We show that these methods agree in results and are more informative and stable than the standard probes.¹

1 Introduction

To estimate to what extent representations (e.g., ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019)) capture a linguistic property, most previous

¹We release code at <https://github.com/lena-voita/description-length-probing>.

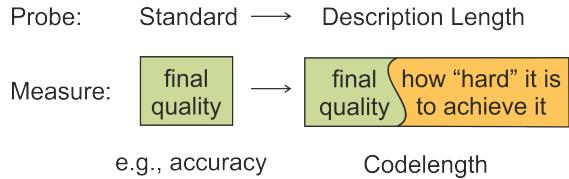


Figure 1: Illustration of the idea behind MDL probes.

work uses ‘probing tasks’ (aka ‘probes’ and ‘diagnostic classifiers’); see Belinkov and Glass (2019) for a comprehensive review. These classifiers are trained to predict a linguistic property from ‘frozen’ representations, and accuracy of the classifier is used to measure how well these representations encode the property.

Despite widespread adoption of such probes, they fail to adequately reflect differences in representations. This is clearly seen when using them to compare pretrained representations with randomly initialized ones (Zhang and Bowman, 2018). Analogously, their accuracy can be similar when probing for genuine linguistic labels and probing for tags randomly associated to word types (‘control tasks’, Hewitt and Liang (2019)). To see differences in the accuracy with respect to these *random baselines*, previous work had to reduce the amount of a probe training data (Zhang and Bowman, 2018) or use smaller models for probes (Hewitt and Liang, 2019).

As an alternative to the standard probing, we take an information-theoretic view at the task of measuring relations between representations and labels. Any regularity in representations with respect to labels can be exploited both to make predictions and to compress these labels, i.e., reduce length of the code needed to transmit them. Formally, we recast learning a model of data (i.e., training a probing classifier) as training it to transmit the data (i.e., labels) in as few bits as possible. This naturally leads to a change of measure: instead of evaluating

probe accuracy, we evaluate *minimum description length (MDL)* of labels given representations, i.e. the minimum number of bits needed to transmit the labels knowing the representations. Note that since labels are transmitted using a model, the model has to be transmitted as well (directly or indirectly). Thus, the overall codelength is a combination of the quality of fit of the model (compressed data length) with the cost of transmitting the model itself.

Intuitively, codelength characterizes not only the final quality of a probe, but also the ‘amount of effort’ needed achieve this quality (Figure 1). If representations have some clear structure with respect to labels, the relation between the representations and the labels can be understood with less effort; for example, (i) the ‘rule’ predicting the label (i.e., the probing model) can be simple, and/or (ii) the amount of data needed to reveal this structure can be small. This is exactly how our vague (so far) notion of ‘the amount of effort’ is translated into codelength. We explain this more formally when describing the two methods for evaluating MDL we use: *variational coding* and *online coding*; they differ in a way they incorporate model cost: directly or indirectly.

Variational code explicitly incorporates cost of transmitting the model (probe weights) in addition to the cost of transmitting the labels; this joint cost is exactly the loss function of a variational learning algorithm (Honkela and Valpola, 2004). As we will see in the experiments, close probe accuracies often come at a very different model cost: the ‘rule’ (the probing model) explaining regularity in the data can be either simple (i.e., easy to communicate) or complicated (i.e., hard to communicate) depending on the strength of this regularity.

Online code provides a way to transmit data without directly transmitting the model. Intuitively, it measures the ability to learn from different amounts of data. In this setting, the data is transmitted in a sequence of portions; at each step, the data transmitted so far is used to understand the regularity in this data and compress the following portion. If the regularity in the data is strong, it can be revealed using a small subset of the data, i.e., early in the transmission process, and can be exploited to efficiently transmit the rest of the dataset. The online code is related to the area under the learning curve, which plots quality as a function of the number of training examples.

If we now recall that, to get reasonable differ-

ences with random baselines, previous work manually tuned (i) model size and/or (ii) the amount of data, we will see that these were indirect ways of accounting for the ‘amount of effort’ component of (i) variational and (ii) online codes, respectively. Interestingly, since variational and online codes are different methods to estimate the same quantity (and, as we will show, they agree in the results), we can conclude that the ability of a probe to achieve good quality using a *small amount of data* and its ability to achieve good quality using a *small probe architecture* reflect the same property: *strength of the regularity in the data*. In contrast to previous work, MDL incorporates this naturally in a theoretically justified way. Moreover, our experiments show that, differently from accuracy, conclusions made by MDL probes are not affected by an underlying probe setting, thus no manual search for settings is required.

We illustrate the effectiveness of MDL for different kinds of random baselines. For example, when considering control tasks (Hewitt and Liang, 2019), while probes have similar accuracies, these accuracies are achieved with a small probe model for the linguistic task and a large model for the random baseline (control task); these architectures are obtained as a byproduct of MDL optimization and not by manual search.

Our contributions are as follows:

- we propose information-theoretic probing which measures MDL of labels given representations;
- we show that MDL naturally characterizes not only probe quality, but also ‘the amount of effort’ needed to achieve it;
- we explain how to easily measure MDL on top of standard probe-training pipelines;
- we show that results of MDL probing are more informative and stable than those of standard probes.

2 Information-Theoretic Viewpoint

Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ be a dataset, where $x_{1:n} = (x_1, x_2, \dots, x_n)$ are representations from a model and $y_{1:n} = (y_1, y_2, \dots, y_n)$ are labels for some linguistic task (we assume that $y_i \in \{1, 2, \dots, K\}$, i.e. we consider classification tasks). As in standard probing task, we want to measure to what extent $x_{1:n}$

encode $y_{1:n}$. Differently from standard probes, we propose to look at this question from the information-theoretic perspective and define the goal of a probe as learning to effectively transmit the data.

Setting. Following the standard information theory notation, let us imagine that Alice has all (x_i, y_i) pairs in \mathcal{D} , Bob has just the x_i 's from \mathcal{D} , and that Alice wants to communicate the y_i 's to Bob. The task is to encode the labels $y_{1:n}$ knowing the inputs $x_{1:n}$ in an optimal way, i.e. with minimal codelength (in bits) needed to transmit $y_{1:n}$.

Transmission: Data and Model. Alice can transmit the labels using some probabilistic model of data $p(y|x)$ (e.g., it can be a trained probing classifier). Since Bob does not know the precise trained model that Alice is using, some explicit or implicit transmission of the model itself is also required. In Section 2.1, we explain how to transmit data using a model p . In Section 2.2, we show direct and indirect ways of transmitting the model.

Interpretation: quality and ‘amount of effort’. In Section 2.3, we show that total codelength characterizes both probe quality and the ‘amount of effort’ needed to achieve it. We draw connections between different interpretations of this ‘amount of effort’ part of the code and manual search for probe settings done in previous work.²

2.1 Transmission of Data Using a Model

Suppose that Alice and Bob have agreed in advance on a model p , and both know the inputs $x_{1:n}$. Then there exists a code to transmit the labels $y_{1:n}$ losslessly with codelength³

$$L_p(y_{1:n}|x_{1:n}) = - \sum_{i=1}^n \log_2 p(y_i|x_i). \quad (1)$$

This is the *Shannon-Huffman code*, which gives an optimal bound on the codelength if the data are independent and come from a conditional probability distribution $p(y|x)$.

Learning is compression. The bound (1) is exactly the categorical cross-entropy loss evaluated on the model p . This shows that the task of compressing labels $y_{1:n}$ is equivalent to learning a

²Note that in this work, we do not consider practical implementations of transmission algorithms; everywhere in the text, ‘codelength’ refers to the theoretical codelength of the associated encodings.

³Up to at most one bit on the whole sequence; for datasets of reasonable size this can be ignored.

model $p(y|x)$: quality of a learned model $p(y|x)$ is the codelength needed to transmit the data.

Compression is usually compared against *uniform encoding* which does not require any learning from data. It assumes $p(y|x) = p_{unif}(y|x) = \frac{1}{K}$, and yields codelength $L_{unif}(y_{1:n}|x_{1:n}) = n \log_2 K$ bits. Another trivial encoding ignores input x and relies on class priors $p(y)$, resulting in codelength $H(y)$.

Relation to Mutual Information. If the inputs and the outputs come from a true joint distribution $q(x, y)$, then, for any transmission method with codelength L , it holds $\mathbb{E}_q[L(y|x)] \geq H(y|x)$ (Grunwald, 2004). Therefore, the gain in codelength over the trivial codelength $H(y)$ is

$$H(y) - \mathbb{E}_q[L(y|x)] \leq H(y) - H(y|x) = I(y; x).$$

In other words, the compression is limited by the mutual information (MI) between inputs (i.e. pre-trained representations) and outputs (i.e. labels).

Note that total codelength includes model codelength in addition to the data code. This means that while high MI is necessary for effective compression, a good representation is the one which also yields simple models predicting y from x , as we formalize in the next section.

2.2 Transmission of the Model (Explicit or Implicit)

We consider two compression methods that can be used with deep learning models (probing classifiers):

- *variational code* – an instance of *two-part* codes, where a model is transmitted explicitly and then used to encode the data;
- *online code* – a way to encode both model and data without *directly* transmitting the model.

2.2.1 Variational Code

We assume that Alice and Bob have agreed on a model class $\mathcal{H} = \{p_\theta | \theta \in \Theta\}$. With *two-part* codes, for any model p_{θ^*} , Alice first transmits its parameters θ^* and then encodes the data while relying on the model. The description length decomposes accordingly:

$$\begin{aligned} L_{\theta^*}^{2\text{-part}}(y_{1:n}|x_{1:n}) &= \\ &= L_{param}(\theta^*) + L_{p_{\theta^*}}(y_{1:n}|x_{1:n}) \\ &= L_{param}(\theta^*) - \sum_{i=1}^n \log_2 p_{\theta^*}(y_i|x_i). \end{aligned} \quad (2)$$

To compute the description length of the parameters $L_{param}(\theta^*)$, we can further assume that Alice and Bob have agreed on a prior distribution over the parameters $\alpha(\theta^*)$. Now, we can rewrite the total description length as

$$-\log_2(\alpha(\theta^*)\epsilon^m) - \sum_{i=1}^n \log_2 p_{\theta^*}(y_i|x_i),$$

where m is the number of parameters and ϵ is a prearranged precision for each parameter. With deep learning models, such straightforward codes for parameters are highly inefficient. Instead, in the variational approach, weights are treated as random variables, and the description length is given by the expectation

$$\begin{aligned} L_{\beta}^{var}(y_{1:n}|x_{1:n}) &= \\ &= -\mathbb{E}_{\theta \sim \beta} \left[\log_2 \alpha(\theta) - \log_2 \beta(\theta) + \sum_{i=1}^n \log_2 p_{\theta}(y_i|x_i) \right] \\ &= KL(\beta \parallel \alpha) - \mathbb{E}_{\theta \sim \beta} \sum_{i=1}^n \log_2 p_{\theta}(y_i|x_i), \end{aligned} \quad (3)$$

where $\beta(\theta) = \prod_{i=1}^m \beta_i(\theta_i)$ is a distribution encoding uncertainty about the parameter values. The distribution $\beta(\theta)$ is chosen by minimizing the codelength given in Expression (3). The formal justification for the description length relies on the *bits-back argument* (Hinton and von Cramp, 1993; Honkela and Valpola, 2004; MacKay, 2003). However, the underlying intuition is straightforward: parameters we are uncertain about can be transmitted at a lower cost as the uncertainty can be used to determine the required precision. The entropy term in Equation (3), $H(\beta) = -\mathbb{E}_{\theta \sim \beta} \log_2 \beta(\theta)$, quantifies this discount.

The negated codelength $-L_{\beta}^{var}(y_{1:n}|x_{1:n})$ is known as the evidence-lower-bound (ELBO) and used as the objective in variational inference. The distribution $\beta(\theta)$ approximates the intractable posterior distribution $p(\theta|x_{1:n}, y_{1:n})$. Consequently, any variational method can in principle be used to estimate the codelength.

In our experiments, we use the network compression method of Louizos et al. (2017). Similarly to variational dropout (Molchanov et al., 2017), it uses sparsity-inducing priors on the parameters, pruning neurons from the probing classifier as a byproduct of optimizing the ELBO. As a result we can assess the probe complexity both using its description length $KL(\beta \parallel \alpha)$ and by inspecting the discovered architecture.

2.2.2 Online (or Prequential) Code

The online (or prequential) code (Rissanen, 1984) is a way to encode both the model and the labels without directly encoding the model weights. In the online setting, Alice and Bob agree on the form of the model $p_{\theta}(y|x)$ with learnable parameters θ , its initial random seeds, and its learning algorithm. They also choose timesteps $1 = t_0 < t_1 < \dots < t_S = n$ and encode data by blocks.⁴ Alice starts by communicating $y_{1:t_1}$ with a uniform code, then both Alice and Bob learn a model $p_{\theta_1}(y|x)$ that predicts y from x using data $\{(x_i, y_i)\}_{i=1}^{t_1}$, and Alice uses that model to communicate the next data block $y_{t_1+1:t_2}$. Then both Alice and Bob learn a model $p_{\theta_2}(y|x)$ from a larger block $\{(x_i, y_i)\}_{i=1}^{t_2}$ and use it to encode $y_{t_2+1:t_3}$. This process continues until the entire dataset has been transmitted. The resulting online codelength is

$$\begin{aligned} L^{online}(y_{1:n}|x_{1:n}) &= t_1 \log_2 K - \\ &\quad - \sum_{i=1}^{S-1} \log_2 p_{\theta_i}(y_{t_i+1:t_{i+1}}|x_{t_i+1:t_{i+1}}). \end{aligned} \quad (4)$$

In this sequential evaluation, a model that performs well with a limited number of training examples will be rewarded by having a shorter codelength (Alice will require fewer bits to transmit the subsequent $y_{t_i:t_{i+1}}$ to Bob). The online code is related to the area under the learning curve, which plots quality (in case of probes, accuracy) as a function of the number of training examples. We will illustrate this in Section 3.2.

2.3 Interpretations of Codelength

Connection to previous work. To get larger differences in scores compared to random baselines, previous work tried to (i) reduce size of a probing model and (ii) reduce the amount of a probe training data. Now we can see that these were indirect ways to account for the ‘amount of effort’ component of (i) variational and (ii) online codes, respectively.

Online code and model size. While the online code does not incorporate model cost explicitly, we can still evaluate model cost by interpreting the difference between the cross-entropy of the model trained on all data and online codelength as the cost of the model. The former is codelength of the data

⁴In all experiments in this paper, the timesteps correspond to 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.25, 12.5, 25, 50, 100 percent of the dataset.

if one knows model parameters, the latter (online codelength) — if one does not know them. In Section 3.2 we will show that trends for model cost evaluated for the online code are similar to those for the variational code. It means that in terms of a code, the ability of a probe to achieve good quality using *small amount of data* or using *a small probe architecture* reflect the same property: the *strength of the regularity in the data*.

Which code to choose? In terms of implementation, the online code uses a standard probe along with its training setting: it trains the probe on increasing subsets of the dataset. Using the variational code requires changing (i) a probing model to a Bayesian model and (ii) the loss function to the corresponding variational loss (3) (i.e. adding the model KL term to the standard data cross-entropy). As we will show later, these methods agree in results. Therefore, the choice of the method can be done depending on the preferences: the variational code can be used to inspect the induced probe architecture, but the online code is easier to implement.

3 Description Length and Control Tasks

Hewitt and Liang (2019) noted that probe accuracy itself does not necessarily reveal if the representations encode the linguistic annotation or if the probe ‘itself’ learned to predict this annotation. They introduced control tasks which associate word types with random outputs, and each word token is assigned its type’s output, regardless of context. By construction, such tasks can only be learned by the probe itself. They argue that selectivity, i.e. difference between linguistic task accuracy and control task accuracy, reveals how much the linguistic probe relies on the regularities encoded in the representations. They propose to tune probe hyperparameters so that to maximize selectivity. In contrast, we will show that MDL probes do not require such tuning.

3.1 Experimental Setting

In all experiments, we use the data and follow the setting of Hewitt and Liang (2019); we build on top of their code and release our extended version to reproduce the experiments.

In the main text, we use a probe with default hyperparameters which was a starting point in Hewitt and Liang (2019) and was shown to have low selectivity. In the appendix, we provide results for

10 different settings and show that, in contrast to accuracy, codelength is stable across settings.

Task: part of speech. Control tasks were designed for two tasks: part-of-speech (PoS) tagging and dependency edge prediction. In this work, we focus only on the PoS tagging task, the task of assigning tags, such as noun, verb, and adjective, to individual word tokens. For the control task, for each word type, a PoS tag is independently sampled from the empirical distribution of the tags in the linguistic data.

Data. The pretrained model is the 5.5 billion-word pre-trained ELMo (Peters et al., 2018). The data comes from Penn Treebank (Marcus et al., 1993) with the traditional parsing training/development/testing splits⁵ without extra preprocessing. Table 1 shows dataset statistics.

Probes. The probe is MLP-2 of Hewitt and Liang (2019) with the default hyperparameters. Namely, it is a multi-layer perceptron with two hidden layers defined as: $y_i \sim \text{softmax}(W_3 \text{ReLU}(W_2 \text{ReLU}(W_1 h_i)))$; hidden layer size h is 1000 and no dropout is used. Additionally, in the appendix, we provide results for both MLP-2 and MLP-1 for several h values: 1000, 500, 250, 100, 50.

For the variational code, we replace dense layers with the Bayesian compression layers from Louizos et al. (2017); the loss function changes to Eq. (3).

Optimizer. All of our probing models are trained with Adam (Kingma and Ba, 2015) with learning rate 0.001. With standard probes, we follow the original paper (Hewitt and Liang, 2019) and anneal the learning rate by a factor of 0.5 once the epoch does not lead to a new minimum loss on the development set; we stop training when 4 such epochs occur in a row. With variational probes, we do not anneal learning rate and train probes for 200 epochs; long training is recommended to enable pruning (Louizos et al., 2017).

3.2 Experimental Results

Results are shown in Table 2.⁶

⁵As given by the code of Qi and Manning (2017) at <https://github.com/qipeng/arc-swift>.

⁶Accuracies can differ from the ones reported in Hewitt and Liang (2019): we report accuracy on the test set, while they – on the development set. Since the development set is used for stopping criteria, we believe that test scores are more reliable.

	Labels	Number of sentences	Number of targets
Part-of-speech	45	39832 / 1700 / 2416	950028 / 40117 / 56684

Table 1: Dataset statistics. Numbers of sentences and targets are given for train / dev / test sets.

	Accuracy		Description Length			
	variational code		online code			
	codelength	compression	codelength	compression		
MLP-2, h=1000						
LAYER 0	93.7 / 96.3	163 / 267	31.32 / 19.09	173 / 302	29.5 / 16.87	
LAYER 1	97.5 / 91.9	85 / 470	59.76 / 10.85	96 / 515	53.06 / 9.89	
LAYER 2	97.3 / 89.4	103 / 612	49.67 / 8.33	115 / 717	44.3 / 7.11	

Table 2: Experimental results; shown in pairs: linguistic task / control task. Codelength is measured in kbits (variational codelength is given in equation (3), online – in equation (4)). Accuracy is shown for the standard probe as in [Hewitt and Liang \(2019\)](#); for the variational probe, scores are similar (see Table 3).

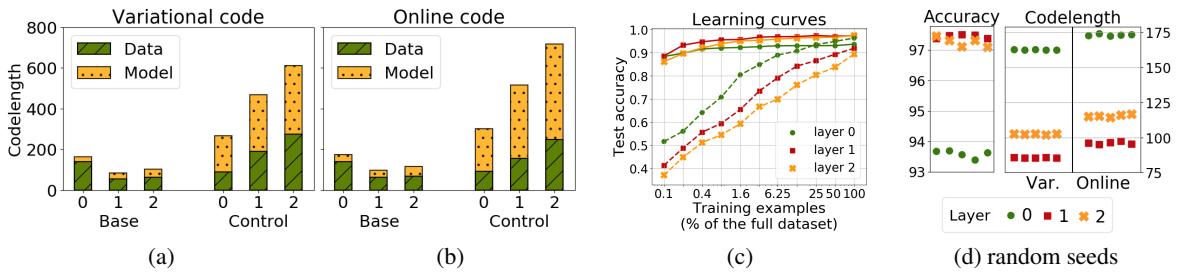


Figure 2: (a), (b): codelength split into data and model codes; (c): learning curves corresponding to online code (solid lines for linguistic task, dashed – for control); (d): results for 5 random seeds, linguistic task (for control task, see appendix).

Different compression methods, similar results.

First, we see that both compression methods show similar trends in codelength. For the linguistic task, the best layer is the first one. For the control task, codes become larger as we move up from the embedding layer; this is expected since the control task measures the ability to memorize word type. Note that codelengths for control tasks are substantially larger than for the linguistic task (at least twice larger). This again illustrates that description length is preferable to probe accuracy: in contrast to accuracy, codelength is able to distinguish these tasks without any search for settings.

LAYER 0: MDL is correct, accuracy is not.

What is even more surprising, *codelength identifies the control task even when accuracy indicates the opposite*: for LAYER 0, accuracy for the control task is higher, but the code is twice longer than for the linguistic task. This is because codelength characterizes how hard it is to achieve this accuracy: for the control task, accuracy is higher, but the cost of achieving this score is very big. We will illustrate

this later in this section.

Embedding vs contextual: drastic difference.

For the linguistic task, note that codelength for the embedding layer is approximately twice larger than that for the first layer. Later in Section 4 we will see the same trends for several other tasks, and will show that even contextualized representations obtained with a randomly initialized model are a lot better than with the embedding layer alone.

Model: small for linguistic, large for control.

Figure 2(a) shows data and model components of the variational code. For control tasks, model size is several times larger than for the linguistic task. This is something that probe accuracy alone is not able to reflect: representations have structure with respect to the linguistic labels and this structure can be ‘explained’ with a small model. The same representations do not have structure with respect to random labels, therefore these labels can be predicted only using a larger model.

Using interpretation from Section 2.3 to split

	Accuracy	Final probe
layer 0		
base	93.5	406-33-49
control	96.3	427-214-137
layer 1		
base	97.7	664-55-35
control	92.2	824-272-260
layer 2		
base	97.3	750-75-41
control	88.7	815-308-481

Table 3: Pruned architecture of a trained variational probe (starting probe: 1024-1000-1000).

the online code into data and model codelength, we get Figure 2(b). The trends are similar to the ones with the variational code; but with the online code, the model component shows how easy it is to learn from small amount of data: if the representations have structure with respect to some labels, this structure can be revealed with a few training examples. Figure 2(c) shows learning curves showing the difference between behavior of the linguistic and control tasks. In addition to probe accuracy, such learning curves have also been used by [Yogatama et al. \(2019\)](#) and [Talmor et al. \(2019\)](#).

Architecture: sparse for linguistic, dense for control. The method for the variational code we use, Bayesian compression of [Louizos et al. \(2017\)](#), lets us assess the induced probe complexity not only by using its description length (as we did above), but also by looking at the induced architecture (Table 3). Probes learned for linguistic tasks are much smaller than those for control tasks, with only 33-75 neurons at the second and third layers. This relates to previous work ([Hewitt and Liang, 2019](#)). The authors considered several pre-defined probe architectures and picked one of them based on a manually defined criterion. In contrast, the variational code gives probe architecture as a byproduct of training and does not need human guidance.

3.3 Stability and Reliability of MDL Probes

Here we discuss stability of MDL results across compression methods, underlying probing classifier setting and random seeds.

The two compression methods agree in results. Note that the observed agreement in codelengths

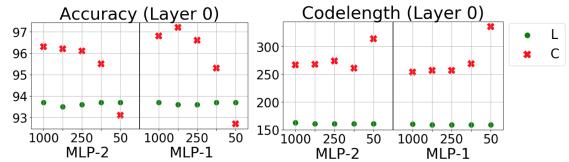


Figure 3: Results for 10 probe settings: accuracy is wrong for 8 out of 10 settings, MDL is always correct (for accuracy higher is better, for codelength – lower).

from different methods (Table 2) is rather surprising: this contrasts to [Blier and Ollivier \(2018\)](#), who experimented with images (MNIST, CIFAR-10) and argued that the variational code yields very poor compression bounds compared to online code. We can speculate that their results may be due to the particular variational approach they use. The agreement between different codes is desirable and suggests sensibility and reliability of the results.

Hyperparameters: change results for accuracy, do not for MDL. While here we will discuss in detail results for the default settings, in the appendix we provide results for 10 different settings; for LAYER 0, results are given in Figure 3. We see that accuracy can change greatly with the settings. For example, difference in accuracy for linguistic and control tasks varies a lot; for LAYER 0 there are settings with contradictory results: accuracy can be higher either for the linguistic or for the control task depending on the settings (Figure 3). In striking contrast to accuracy, *MDL results are stable across settings*, thus *MDL does not require search for probe settings*.

Random seed: affects accuracy but not MDL. We evaluated results from Table 2 for random seeds from 0 to 4; for the linguistic task, results are shown in Figure 2(d). We see that using accuracy can lead to different rankings of layers depending on a random seed, making it hard to draw conclusions about their relative qualities. For example, accuracy for LAYER 1 and LAYER 2 are 97.48 and 97.31 for seed 1, but 97.38 and 97.48 for seed 0. On the contrary, the MDL results are stable and the scores given to different layers are well separated.

Note that for this ‘real’ task, where the true ranking of layers 1 and 2 is not known in advance, tuning a probe setting by maximizing difference with the synthetic control task (as done by [Hewitt and Liang \(2019\)](#)) does not help: in the tuned setting, scores for these layers remain very close (e.g., 97.3 and 97.0 ([Hewitt and Liang, 2019](#))).

Part-of-speech	I want to find more , [something] bigger or deeper . → NN (Noun)
Constituents	I want to find more , [something bigger or deeper] . → NP (Noun Phrase)
Dependencies	[I] ₁ am not [sure] ₂ how reliable that is , though . → nsubj (nominal subject)
Entities	The most fascinating is the maze known as [Wind Cave] . → LOC
SRL	I want to [find] ₁ [more , something bigger or deeper] ₂ . → Agr1 (Agent)
Coreference	So [the followers] ₁ waited to say anything about what [they] ₂ saw . → True
Rel. (SemEval)	The [shaman] ₁ cured him with [herbs] ₂ . → Instrument-Agency(e2, e1)

Table 4: Examples of sentences, spans, and target labels for each task.

	Labels	Number of sentences	Number of targets
Part-of-speech	48	115812 / 15680 / 12217	2070382 / 290013 / 212121
Constituents	30	115812 / 15680 / 12217	1851590 / 255133 / 190535
Dependencies	49	12522 / 2000 / 2075	203919 / 25110 / 25049
Entities	18	115812 / 15680 / 12217	128738 / 20354 / 12586
SRL	66	253070 / 35297 / 26715	598983 / 83362 / 61716
Coreference	2	115812 / 15680 / 12217	207830 / 26333 / 27800
Rel. (SemEval)	19	6851 / 1149 / 2717	6851 / 1149 / 2717

Table 5: Dataset statistics. Numbers of sentences and targets are given for train / dev / test sets.

4 Description Length and Random Models

Now, from random labels for word types, we come to another type of random baselines: randomly initialized models. Probes using these representations show surprisingly strong performance for both token (Zhang and Bowman, 2018) and sentence (Wieting and Kiela, 2019) representations. This again confirms that accuracy alone does not reflect what a representation encodes. With MDL probes, we will see that codelength shows large difference between trained and randomly initialized representations.

In this part, we also experiment with ELMo and compare it with a version of the ELMo model in which all weights above the lexical layer (LAYER 0) are replaced with random orthonormal matrices (but the embedding layer, LAYER 0, is retained from trained ELMo). We conduct a line of experiments using a suite of edge probing tasks (Tenney et al., 2019). In these tasks, a probing model (Figure 4) can access only representations within given spans, such as a predicate-argument pair, and must predict properties, such as semantic roles.

4.1 Experimental Setting

Tasks and datasets. We focus on several core NLP tasks: PoS tagging, syntactic constituent and dependency labeling, named entity recognition, se-

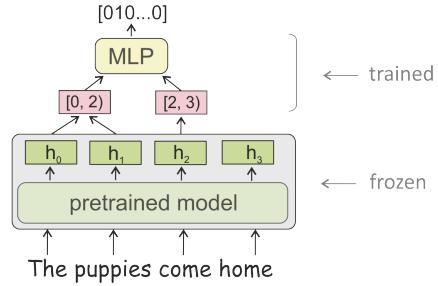


Figure 4: Probing model architecture for an edge probing task. The example is for semantic role labeling; for PoS, NER and constituents, only a single span is used.

mantic role labeling, coreference resolution, and relation classification. Examples for each task are shown in Table 4, dataset statistics are in Table 5. See extra details in Tenney et al. (2019).

We follow Tenney et al. (2019) and use ELMo (Peters et al., 2018) trained on the Billion Word Benchmark dataset (Chelba et al., 2014).

Probes and optimization. Probing architecture is illustrated in Figure 4. It takes a list of contextual vectors $[e_0, e_1, \dots, e_n]$ and integer spans $s_1 = [i_1, j_1]$ and (optionally) $s_2 = [i_2, j_2]$ as inputs, and uses a projection layer followed by the self-attention pooling operator of Lee et al. (2017) to compute fixed-length span representations. The span representations are concatenated and fed into a two-layer MLP followed by a softmax output

	Accuracy		Description Length			
			variational code		online code	
	codelength	compression	codelength	compression	codelength	compression
Part-of-speech						
L0	91.3	483	23.4	462	24.5	
L1	97.8 / 95.7	209 / 273	54.0 / 41.4	192 / 294	58.8 / 38.5	
L2	97.5 / 95.7	252 / 273	44.7 / 41.4	216 / 294	52.3 / 38.5	
Constituents						
L0	75.9	1181	7.5	1149	7.7	
L1	86.4 / 77.6	603 / 877	14.7 / 10.1	570 / 1081	15.6 / 8.2	
L2	85.1 / 77.6	719 / 875	12.3 / 10.1	680 / 1074	13.1 / 8.3	
Dependencies						
L0	80.9	158	7.1	175	6.4	
L1	94.0 / 90.3	80 / 103	14.0 / 10.8	74 / 106	15.1 / 10.5	
L2	92.8 / 90.4	94 / 103	11.9 / 10.8	82 / 106	13.7 / 10.6	
Entities						
L0	92.3	40	13.2	40	13.1	
L1	95.0 / 93.5	27 / 34	19.3 / 15.4	27 / 35	19.8 / 15.1	
L2	95.3 / 93.6	30 / 34	17.7 / 15.2	26 / 35	19.9 / 15.1	
SRL						
L0	81.1	411	8.6	381	9.3	
L1	91.9 / 84.4	228 / 306	15.5 / 11.5	212 / 365	16.7 / 9.7	
L2	90.2 / 84.5	272 / 306	13.0 / 11.6	245 / 363	14.4 / 9.7	
Coreference						
L0	89.9	57.4	3.54	60	3.4	
L1	92.9 / 90.7	50.3 / 54.5	4.04 / 3.72	51 / 65	4.0 / 3.1	
L2	92.2 / 90.4	56.8 / 54.3	3.57 / 3.74	55 / 65	3.7 / 3.1	
Rel. (SemEval)						
L0	55.8	11.5	2.48	15.9	1.79	
L1	75.2 / 69.1	8.0 / 9.7	3.56 / 2.94	8.8 / 11.8	3.2 / 2.4	
L2	77.0 / 68.9	8.4 / 9.7	3.40 / 2.92	8.6 / 11.7	3.3 / 2.4	

Table 6: Experimental results; shown in pairs: trained model / randomly initialized model. Codelength is measured in kbits (variational codelength is given in equation (3), online – in equation (4)), compression – with respect to the corresponding uniform code.

layer. As in the original paper, we use the standard cross-entropy loss, hidden layer size of 256 and dropout of 0.3. For further details on training, we refer the reader to the original paper by Tenney et al. (2019).⁷

For the variational code, the layers are replaced with that of Bayesian compression by Louizos et al. (2017); loss function changes to (3) and no dropout

⁷The differences with the original implementation by Tenney et al. (2019) are: softmax with the cross-entropy loss instead of sigmoid with binary cross-entropy, using the loss instead of F1 in the early stopping criterion.

is used. Similar to the experiments in the previous section, we do not anneal learning rate and train at least 200 epochs to enable pruning.

We build our experiments on top of the original code by Tenney et al. (2019) and release our extended version.

4.2 Experimental Results

Results are shown in Table 6.

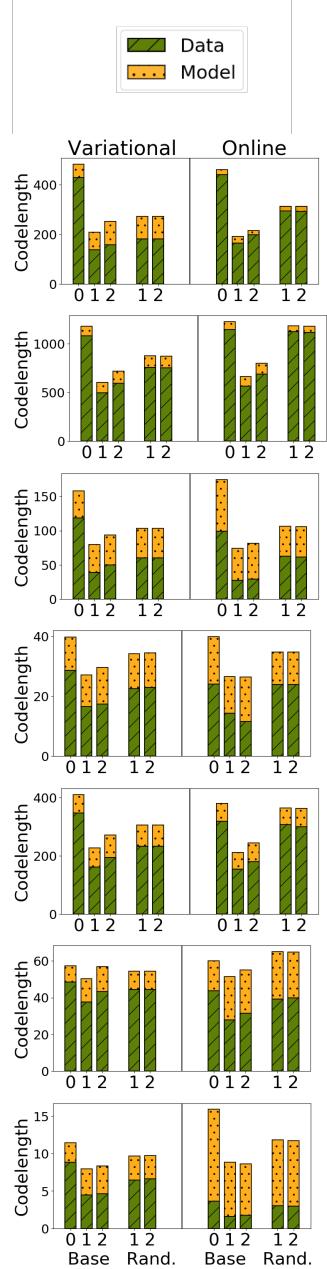


Table 7: Data and model code components for the tasks from Table 6.

tic difference between the embedding layer (LAYER 0) and contextualized representations: codelengths differ about twice for most of the tasks. Both compression methods show that even for the randomly initialized model, contextualized representations are better than lexical representations. This is because context-agnostic embeddings do not contain enough information about the task, i.e., MI between labels and context-agnostic representations is smaller than between labels and contextualized representations. Since compression of the labels given model (i.e., data component of the code) is limited by the MI between the representations and the labels (Section 2.1), the data component of the codelength is much bigger for the embedding layer than for contextualized representations.

Trained vs random. As expected, codelengths for the randomly initialized model are larger than for the trained one. This is more prominent when not just looking at the bare scores, but comparing compression against context-agnostic representations. For all tasks, compression bounds for the randomly initialized model are closer to those of context-agnostic LAYER 0 than representations from the trained model. This shows that gain from using context for the randomly initialized model is at least twice smaller than for the trained model.

Note also that *randomly initialized layers do not evolve*: for all tasks, MDL for layers of the randomly initialized model is the same. Moreover, Table 7 shows that not only total codelength but data and model components of the code for random model layers are also identical. For the trained model, this is not the case: LAYER 2 is worse than LAYER 1 for all tasks. This is one more illustration of the general process explained in Voita et al. (2019a): the way representations evolve between layers is defined by the training objective. For the randomly initialized model, since no training objective has been optimized, no evolution happens.

5 Related work

Probing classifiers are the most common approach for associating neural network representations with linguistic properties (see Belinkov and Glass (2019) for a survey). Among the works highlighting limitations of standard probes (not mentioned earlier) is the work by Saphra and Lopez (2019), who show that diagnostic classifiers are not suitable for understanding learning dynamics.

In addition to task performance, learning curves

have also been used before by Yogatama et al. (2019) to evaluate how quickly a model learns a new task, and by Talmor et al. (2019) to understand whether the performance of a LM on a task should be attributed to the pre-trained representations or to the process of fine-tuning on the task data.

Other methods for analyzing NLP models include (i) inspecting the mechanisms a model uses to encode information, such as attention weights (Voita et al., 2018; Raganato and Tiedemann, 2018; Voita et al., 2019b; Clark et al., 2019; Kovaleva et al., 2019) or individual neurons (Karpasathy et al., 2015; Pham et al., 2016; Bau et al., 2019), (ii) looking at model predictions using manually defined templates, either evaluating sensitivity to specific grammatical errors (Linzen et al., 2016; Gulordava et al., 2018; Tran et al., 2018; Marvin and Linzen, 2018) or understanding what language models know when applying them as knowledge bases or in question answering settings (Radford et al., 2019; Petroni et al., 2019; Poerner et al., 2019; Jiang et al., 2019).

An information-theoretic view on analysis of NLP models has been previously attempted in Voita et al. (2019a) when explaining how representations in the Transformer evolve between layers under different training objectives.

6 Conclusions

We propose information-theoretic probing which measures *minimum description length (MDL)* of labels given representations. We show that MDL naturally characterizes not only probe quality, but also ‘the amount of effort’ needed to achieve it (or, intuitively, *strength of the regularity* in representations with respect to the labels); this is done in a theoretically justified way without manual search for settings. We explain how to easily measure MDL on top of standard probe-training pipelines. We show that results of MDL probing are more informative and stable compared to the standard probes.

Acknowledgments

IT acknowledges support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

References

- Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. [Identifying and controlling important neurons in neural machine translation](#). In *International Conference on Learning Representations*, New Orleans.
- Yonatan Belinkov and James Glass. 2019. [Analysis methods in neural language processing: A survey](#). *Transactions of the Association for Computational Linguistics*, 7:49–72.
- Léonard Blier and Yann Ollivier. 2018. [The description length of deep learning models](#). In *Advances in Neural Information Processing Systems*, pages 2216–2226.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Philipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. [What does BERT look at? an analysis of BERT’s attention](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Peter Grunwald. 2004. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*.
- Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. [Colorless green recurrent networks dream hierarchically](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.
- John Hewitt and Percy Liang. 2019. [Designing and interpreting probes with control tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.
- GE Hinton and D von Cramp. 1993. Keeping neural networks simple by minimising the description length of weights. In *Proceedings of COLT-93*, pages 5–13.
- Antti Honkela and Harri Valpola. 2004. [Variational learning and bits-back coding: an information-theoretic view to bayesian learning](#). In *IEEE Transactions on Neural Networks*, volume 15, pages 800–810.
- Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2019. How can we know what language models know? *arXiv preprint arXiv:1911.12543*.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representation (ICLR 2015)*.
- Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. [Revealing the dark secrets of BERT](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.
- Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. [End-to-end neural coreference resolution](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.
- Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. [Assessing the ability of lstms to learn syntax-sensitive dependencies](#). *Transactions of the Association for Computational Linguistics*, 4:521–535.
- Christos Louizos, Karen Ullrich, and Max Welling. 2017. [Bayesian compression for deep learning](#). In *Advances in Neural Information Processing Systems*, pages 3288–3298.
- David JC MacKay. 2003. *Information theory, inference and learning algorithms*. Cambridge university press.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. [Building a large annotated corpus of English: The Penn Treebank](#). *Computational Linguistics*, 19(2):313–330.
- Rebecca Marvin and Tal Linzen. 2018. [Targeted syntactic evaluation of language models](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

- Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.
- Ngoc-Quan Pham, German Kruszewski, and Gemma Boleda. 2016. Convolutional neural network language models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1153–1162, Austin, Texas. Association for Computational Linguistics.
- Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *arXiv preprint arXiv:1911.03681*.
- Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117, Vancouver, Canada. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.
- Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.
- Jorma Rissanen. 1984. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information theory*, 30(4):629–636.
- Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019. olmpics – on what language model pre-training captures. *arXiv preprint arXiv:1912.13283*.
- Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.
- Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736, Brussels, Belgium. Association for Computational Linguistics.
- Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.
- Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.
- John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *International Conference on Learning Representations*.
- Dani Yogatama, Cyprien de Masson d’Autume, Jerome Connor, Tomas Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.
- Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does:

Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.

A Description Length and Control Tasks

A.1 Settings

Results are given in Table 8.

Accuracy	Description Length				
	variational code		online code		
	codelength	compr.	codelength	compr.	
MLP-2, h=1000					
L 0	93.7 / 96.3	163 / 267	32 / 19	173 / 302	30 / 17
L 1	97.5 / 91.9	85 / 470	60 / 11	96 / 515	53 / 10
L 2	97.3 / 89.4	103 / 612	50 / 8	115 / 717	44 / 7
MLP-2, h=500					
L 0	93.5 / 96.2	161 / 268	32 / 19	170 / 313	30 / 16
L 1	97.8 / 92.1	84 / 470	61 / 11	93 / 547	55 / 9
L 2	97.1 / 86.5	102 / 611	50 / 8	112 / 755	46 / 7
MLP-2, h=250					
L 0	93.6 / 96.1	161 / 274	32 / 19	169 / 328	30 / 16
L 1	97.7 / 90.3	84 / 470	61 / 11	91 / 582	56 / 9
L 2	97.1 / 85.2	101 / 611	50 / 8	112 / 799	46 / 6
MLP-2, h=100					
L 0	93.7 / 95.5	161 / 261	32 / 20	167 / 367	31 / 14
L 1	97.6 / 86.9	84 / 492	61 / 10	91 / 678	56 / 8
L 2	97.2 / 80.9	102 / 679	50 / 8	112 / 901	46 / 6
MLP-2, h=50					
L 0	93.7 / 93.1	161 / 314	32 / 16	166 / 416	31 / 12
L 1	97.6 / 82.7	84 / 605	61 / 8	93 / 781	55 / 7
L 2	97.0 / 76.2	102 / 833	50 / 6	116 / 1007	44 / 5
MLP-1, h=1000					
L 0	93.7 / 96.8	160 / 254	32 / 20	166 / 275	31 / 19
L 1	97.7 / 92.7	82 / 468	62 / 11	88 / 477	58 / 11
L 2	97.0 / 86.7	100 / 618	51 / 8	107 / 696	48 / 7
MLP-1, h=500					
L 0	93.6 / 97.2	159 / 257	32 / 20	164 / 295	31 / 17
L 1	97.5 / 91.6	82 / 468	62 / 11	88 / 516	58 / 10
L 2	97.0 / 86.3	100 / 619	51 / 8	107 / 736	48 / 7
MLP-1, h=250					
L 0	93.6 / 96.6	159 / 257	32 / 20	164 / 316	31 / 16
L 1	97.5 / 89.9	82 / 473	62 / 11	87 / 574	58 / 9
L 2	97.1 / 84.2	99 / 632	51 / 8	109 / 795	47 / 6
MLP-1, h=100					
L 0	93.7 / 95.3	159 / 269	32 / 19	163 / 374	31 / 14
L 1	97.6 / 86.4	82 / 525	62 / 10	87 / 683	58 / 8
L 2	97.1 / 80.0	100 / 731	51 / 7	109 / 905	47 / 6
MLP-1, h=50					
L 0	93.7 / 92.7	159 / 336	32 / 15	164 / 438	31 / 11
L 1	97.6 / 82.0	82 / 648	62 / 8	90 / 790	56 / 7
L 2	97.2 / 75.0	100 / 875	51 / 6	114 / 1016	45 / 5

Table 8: Experimental results; shown in pairs: linguistic task / control task. Codelength is measured in kbits (variational codelength is given in equation (3), online – in equation (4)). h is the probe hidden layer size.

A.2 Random seeds: control task

Results are shown in Figure 5.

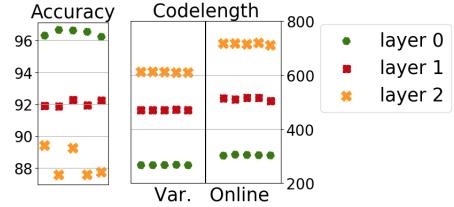


Figure 5: Results for 5 random seeds, control task (default setting: MLP-2, $h = 1000$).

B Description Length and Random Models

	Accuracy	Final probe
layer 0		
base	91.31	728-31-154
layer 1		
base	97.7	878-42-172
random	96.76	876-50-228
layer 2		
base	97.32	872-50-211
random	96.76	929-47-229

Table 9: Pruned architecture of a trained variational probe, Part of Speech (starting probe: 1024-256-256).

	Accuracy	Final probe
layer 0		
base	75.61	976-47-242
layer 1		
base	86.01	1011-53-227
random	81.35	1001-57-235
layer 2		
base	84.36	985-61-238
random	81.42	971-57-234

Table 10: Pruned architecture of a trained variational probe, constituent labeling (starting probe: 1024-256-256).

	Accuracy	Final probe
layer 0		
base	80.11	(423+356)-36-119
layer 1		
base	92.3	(682+565)-38-85
random	89.86	(635+548)-40-98
layer 2		
base	90.6	(581+422)-42-104
random	89.96	(646+538)-38-94

Table 11: Pruned architecture of a trained variational probe, dependency labeling (starting probe: (1024+1024)-512-256).

	Accuracy	Final probe
layer 0		
base	91.7	450-16-36
layer 1		
base	94.95	509-16-35
random	93.36	551-18-36
layer 2		
base	94.93	527-17-41
random	93.57	536-18-34

Table 12: Pruned architecture of a trained variational probe, named entity recognition (starting probe: 1024-256-256).

	Accuracy	Final probe
layer 0		
base	79.1	(567+754)-46-158
layer 1		
base	90.25	(709+937)-48-140
random	86.59	(678+857)-55-148
layer 2		
base	88.5	(601+863)-52-142
random	86.34	(744+889)-53-151

Table 13: Pruned architecture of a trained variational probe, semantic role labeling (starting probe: (1024+1024)-512-256).

	Accuracy	Final probe
layer 0		
base	88.87	(358+352)-16-20
layer 1		
base	91.6	(497+492)-20-22
random	90.35	(363+357)-23-21
layer 2		
base	90.29	(519+505)-18-19
random	90.45	(375+377)-21-21

Table 14: Pruned architecture of a trained variational probe, coreference resolution (starting probe: (1024+1024)-512-256).

	Accuracy	Final probe
layer 0		
base	48.77	(138+137)-10-14
layer 1		
base	71.07	(116+178)-16-17
random	60.73	(168+135)-15-15
layer 2		
base	71.59	(123+164)-14-18
random	60.69	(167+125)-13-15

Table 15: Pruned architecture of a trained variational probe, relation classification (starting probe: (1024+1024)-512-256).