more invariant second-layer features. The corresponding model is denoted CAE-2_p.

How to measure invariance to transformations known a-priori? In order to validate that the proposed criterion does learn features that are more invariant to transformations of interest, we define the following *average normalized sensitivity* score, following the ideas put forward in Goodfellow et al. (2009). Let $T(x)$ represent a random deformation of x in directions of variability known a priori, such as affine transformations of the ink in an image (corresponding for example to small translations, rotations or scaling of the content in the image). Then $(f_i(x) - f_i(T(x)))^2$ measures how sensitive is unit i to T (when large) or how invariant to it (when small) it is. Following Goodfellow et al. (2009) we also need some form of normalization to account for features that do not vary much (maybe even constant): we will normalize the sensitivity of each unit by $V[f_i]$, the empirical variance of $f_i(x)$ over the data set. This yields for each unit i and each input x a normalized sensitivity

$$s_i(x) = \frac{(f_i(x) - f_i(T(x)))^2}{V[f_i]}.$$

Like Goodfellow et al. (2009), we focus on a fraction of the units, here the 20% with highest variance, and define the normalized sensitivity $\gamma(x)$ of the *layer* for an example x as the average of the normalized sensitivity of these selected units. Finally, the *average normalized sensitivity* score $\bar{\gamma}$ is obtained by computing the average of $\gamma(x)$ over the training set. When comparing layers learned by two different models A and B, statistical significance of the difference between $\bar{\gamma}_A$ and $\bar{\gamma}_B$ is assessed by computing the standard error[4] of the mean of differences $\gamma_A(x) - \gamma_B(x)$.

Experimental comparison of sensitivity to a-priori known deformations. We considered stochastic affine deformation $T(x)$ for MNIST test-set digits, controlled by a set of 6 random parameters. These produced slightly shifted, scaled, rotated or slanted variations of the digits.

Table 2 compares the resulting *average normalized sensitivity* score obtained by the second layer learned by the CAE-2_p algorithm, to that obtained for several alternative models. These results confirm that the CAE-2_p has learned features that are significantly more invariant to this kind of deformations, even though they have not been explicitly used during training. DBN-2 is a Deep Belief Network obtained by stacking two RBMs.

Table 2. Average normalized sensitivity ($\bar{\gamma}$) of last layer to affine deformations of MNIST digits. The deformations were not used in any way during training. Second column shows difference with CAE-2_p together with standard error of the differences. The proposed CAE-2_p appears to be *significantly* less sensitive (more invariant) than other models.

Model	$\bar{\gamma}$	$\bar{\gamma} - \bar{\gamma}_{CAE-2_p}$	
CAE-1	8.23	6.28	± 0.04
CAE-2	2.84	0.89	± 0.08
DBN-2	2.36	0.41	± 0.025
CAE-2_p	**1.95**	0	

Effect of invariant-feature learning on classification performance. Next we wanted to check whether the ability to learn more invariant features of the CAE-2_p could translate to better classification performance. Table 3 shows classification performance on the TFD dataset of a deep neural network pretrained using several variants of CAEs, and then fine-tuned on the supervised classification task. For comparison we also give the best result we obtained with a non-pretrained multi-layer perceptron (MLP). We see that the criterion for learning more invariant features used in CAE-2_p yields a significant improvement in classification. We get the best performance among methods that do not explicitly use prior domain knowledge.[5]

Table 3. Test classification error of several models, trained on TFD, averaged over 5 folds (reported with standard deviation).

MLP	CAE-1	CAE-2	CAE-2_p
26.17 ± 3.06	24.12 ± 1.87	23.73 ± 1.62	**21.78** ± 1.04

6. Future Work and Conclusion

We have proposed a converging stochastic process which exploits what has been learned by a CAE to generate samples, and we have found that these samples are not only visually good, they "generalize" well, in the sense of populating the space in the same areas where one tends to find test examples. A related idea has also allowed us to train the second layer of a 2-layer CAE, acting like pooling or invariance-seeking features, and this has yielded improvements in classi-

[4] This is a slight approximation that considers the $V[f_i]$ as constants.

[5] for comparison, Ranzato et al. (2011) reports that SVM achieve 28.5% and sparse coding : 23.4%. Better performance is obtained by convolutional architectures (17.6%) but the convolutional architecture and hard-coded pooling uses prior knowledge of the topology of images.