An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems

Rainer Kelz ∗1 and Gerhard Widmer1

1Department of Computational Perception, Johannes Kepler University Linz, Austria

February 2, 2017

Abstract

Several recent polyphonic music transcription systems have utilized deep neural networks to achieve state of the art results on various benchmark datasets, pushing the envelope on framewise and note-level performance measures. Unfortunately we can observe a sort of glass ceiling effect. To investigate this effect, we provide a detailed analysis of the particular kinds of errors that state of the art deep neural transcription systems make, when trained and tested on a piano transcription task. We are ultimately forced to draw a rather disheartening conclusion: the networks seem to learn combinations of notes, and have a hard time generalizing to unseen combinations of notes. Furthermore, we speculate on various means to alleviate this situation.

1 Introduction

The problem of polyphonic transcription can be formally described as the transformation of a time-ordered sequence of (audio) samples \( \mathbf{X} = (x)_t^{T}, x_t \in \mathcal{X} \) into a set of tuples \((t_s, t_e, F_0, A)\), describing start, end, fundamental frequency or pitch and optionally amplitude of the notes that were played. A slightly easier problem is framewise transcription, or tone-quantized multi-\( F_0 \) estimation, where the output is a time-ordered sequence \( \mathbf{Y} = (y)_t^{T}, y_t \in \mathcal{Y} \), with \( y_t \in \{0,1\}^K \) being a vector of indicator variables and \( K \) denoting the tonal range. In other words, the \( y \) vectors specify the note pitches believed to be active in a given audio frame \( x \). Another simplifying assumption is usually the presence of only a single instrument, which more often than not turns out to be the piano, having a tonal range of \( K = 88 \).

We will focus on framewise transcription systems only, as they turn out to be a crucial stage in the full transcription process, especially in so called hybrid systems that post-process the framewise output with dynamic probabilistic models to extract the aforementioned tuples describing musical notes, such as [13, 14].

A diverse set of methods have been employed to tackle the framewise transcription problem, with non-negative matrix factorization being one of the more prominent methods. Smaragdis and Brown with their seminal paper [15] using non-negative matrix factorization (NMF) for polyphonic transcription already identified an undesirable property of the technique. NMF seeks to minimize the reconstruction error \( \| \mathbf{X} - \mathbf{WH} \|_N \), where \( \mathbf{X} \in \mathbb{R}^{D \times T} \) is the vector valued signal to reconstruct, \( \mathbf{W} \in \mathbb{R}^{D \times d} \) is the dictionary, \( \mathbf{H} \in \mathbb{R}^{d \times T} \) are the activations in time of the bases and \( N(\cdot) \) is a matrix norm. If no additional constraints are applied and no a priori knowledge is exploited, Smaragdis and Brown [13] note that the method learns a dictionary of unique events, rather than individual notes. Two remedies for this problem are also named: either choose sets of notes in

∗Address correspondence to: rainer.kelz@jku.at
such a way that from their intersection single notes can be identified, or present all individual notes in isolation, so a meaningful dictionary can be learned first. A similar effect is achievable if the dictionary matrix is harmonically constrained. The latter two methods seem to be popular choices in the literature \cite{15, 1, 2, 3, 4, 6, 11, 16, 17, 8} to solve this problem for NMF.

We conduct a simple experiment to examine whether neural networks trained for a piano transcription task suffer from the same disentanglement problems, followed by an analysis of two very different neural network architectures and the extent to which they exhibit this behavior.

2 Methods

Lacking proper theoretical analytic tools for the model class of neural networks, we resort to empirical tools, namely computational experiments. We train several deep neural networks in a supervised fashion for a framewise piano transcription task and analyze their error behavior. Adhering very closely to already established model architectures, as exemplified in \cite{14, 7}, we deviate only in very few aspects, mostly concerning hyperparameter choices that affect training time but have little effect on performance. These parametrized functions we learn are of the following form: $f_{\text{net}}: \mathcal{X} \rightarrow \mathcal{Y}$, with $f_{\text{net}}$ in turn being composed of multiple simpler functions, commonly referred to as layers in the neural network literature. An example of a network with an input, hidden and output layer would be $f_{\text{net}}(x) = f_3(f_2(f_1(x; \theta_3); \theta_2); \theta_3)$, where $f_i(z_{i-1}; \theta_i) = \sigma(W_i z_{i-1} + b_i)$ with $\theta_i = \{W_i, b_i\}$ having matching dimensions to fit the output $z_{i-1}$ of the previous layer. $\sigma(\cdot)$ is a nonlinear function applied elementwise. We note here that the functions $f_i$ may actually have more than one input $z$, and it may also be from layers other than the directly previous layer. We do not explicitly model convolution as it can be expressed as a matrix-matrix product, given $W$ and $z$ have the right shapes.

We choose neural network architectures already established to work well for framewise transcription. Our first choice is exactly the ConvNet architecture as proposed in \cite{7}, which achieves state of the art results for framewise transcription on a popular benchmark dataset. We also designed a much smaller version of this network, which will be referred to as SmallConvNet. Additionally, we borrow an architecture originally employed for medical image segmentation, called the UNet \cite{12} and make two small modifications to adapt it for our purposes. We call the adapted architecture $AUNet$. It is able to directly integrate information at different scales, which is beneficial for smoothing in the temporal direction, and identifying groups of partials and their distance in the frequency dimension. The precise definitions for all networks, as well as schematic drawings of the architectures for the ConvNet, the SmallConvNet and the $AUNet$ can be found in the appendix. The definitions are listed in tables 1, 2 and 3 whereas the schemata are depicted in figures 7a, 7b and 8 respectively.

3 Datasets

We use a synthetic dataset to conduct small scale experiments with the SmallConvNet. A subset of the MAPS dataset \cite{5} is used to train and test the ConvNet and the $AUNet$ models. The MAPS dataset consists of several classical piano pieces, along with isolated notes and common chords, rendered with 7 different software synthesizers (samplers) in addition to 2 Disklavier piano recordings, one with the microphone close to the piano, and one with the microphone farther away and thus containing room acoustics. We now describe each subset in turn:

3.1 FLUID

For focused computational experiments we synthesize two-note combinations and isolated notes. We only use notes within an 11 semitone range around a reference pitch (C4/MIDI60), creating $\binom{25}{2} = 253$ two-note intervals. The onset and offset of the two notes are exactly synchronous. We use the free software sampler Fluidsynth\footnote{www.fluidsynth.org} together with the freely available
Fluid-R3-GM\footnote{http://www.musescore.org/download/fluid-soundfont.tar.gz} soundfont to render a dataset FLUID-COMBI where the train and validation sets both consist of the aforementioned intervals, whereas the test set contains individual notes only. For FLUID-ISOL, the individual notes are in the train and validation sets, whereas the test set contains the intervals. So for both datasets the intersection of unique events in train and test sets is the empty set. The error behavior of the SmallConvNet on this dataset is discussed in section 4.1.

3.2 MAPS-MUS

This subset consists only of the rendered classical piano pieces in the MAPS dataset. We adopt the more realistic train-test scenario described in \cite{14}, which is referred to as Configuration-II. It is more realistic because it trains only on synthetic renderings, and tests on the real piano recordings. We select the training set as all pieces from 6 synthesizers, the validation set is comprised of all renderings from a randomly selected 7th, and the test set is made up of all Disklavier recordings. We will refer to this dataset as MAPS-MUS from now on. The respective error behaviors of the two larger models, the ConvNet and the AUNet on this dataset are discussed in section 4.2. We did not use the test set for conducting any error analysis, other than measuring final performance after model selection, to make sure that both models actually achieve state of the art results. The rationale behind this is explained in detail in section 4.2.

4 Results

4.1 SmallConvNet and FLUID

We start with a controlled empirical analysis of the disentanglement problem using our synthetic datasets. We train the SmallConvNet for framewise transcription on logarithmic filtered, log-magnitude spectrograms with 229 bins, as proposed in \cite{7}. The output size of the network is limited to 23 notes, and it has only 5327 parameters, to make it approximately comparable to NMF with a dictionary matrix $W \in \mathbb{R}^{229 \times 23}$ having 5267 parameters. We note that overfitting, fitting noise in the data, is not the real problem here, as the acoustic properties of the sound sources are the same for train and test set.

The general idea of this experiment is discovering to which extent the network is capable of detecting isolated notes, if all it has ever seen were combinations, and vice versa. We can find a partial answer to this question in figures 1 and 2. The figures all show the proportion of frames where all notes have been exactly identified, and contrast them with the proportion of frames in which notes have been added or omitted. This means that the three quantities do not necessarily sum to one, because notes could have been added and some others omitted in a frame.

In figure 1 we can observe that after seeing only two-note intervals, the network is able to generalize to isolated notes to some extent. While a surprising number of individual notes are transcribed perfectly, some notes are still not recognized properly. For these notes their companion notes from the train set are predicted as simultaneously sounding, indicating a failure to disentangle note combinations during training.

Figure 1: For isolated notes present only in the test set, this is the proportion of exactly transcribed frames, along with the proportions of frames that had notes added or omitted, respectively. Transcriptions stem from the SmallConvNet trained on FLUID-COMBI.

In figure 2 we see that the network utterly fails to generalize from isolated notes to note combinations, with only two exceptions. We plotted only the 23 best
transcribed note combinations, as for the remaining 230 intervals the proportion of omission errors is very close to or even exactly 1.0. The network does manage to transcribe two of the intervals with acceptable accuracy, however an explanation of why exactly these two intervals could be recognized eludes us at the moment.

We might draw a preliminary conclusion from these results: the strategy most successful for alleviating the disentanglement problem for NMF, namely learning the dictionary $W$ from isolated notes, does not work for neural transcription systems. The NMF of spectrograms is a linear system, and therefore has the superposition property. Its response to multiple inputs is the sum of the responses for individual inputs. This is not necessarily true for neural networks, as they may learn to approximate a linear function, but do not have to.

The other strategy mentioned in [15], namely showing combinations of notes to the networks, seems to work fairly well for the majority of isolated notes, as can be observed in figure [1]. Unfortunately, the number of combinations for the tonal range of the piano grows large very quickly. Even when assuming a maximum polyphony of only 6, we would already need to show $\sum_{i=2}^{6} \binom{88}{i} = 41,621,206$ combinations to the network.

4.2 **ConvNet, AUNet and MAPS-MUS**

We now turn our attention to a more musically relevant dataset. We train several instances of both a ConvNet and an AUNet, closely adhering to the training procedure described in [2], and select the model for analysis that achieves highest framewise f-measure on the validation set. Our analysis of error behavior is restricted to the validation set as well, simply because we want to avoid learning too much about the composition of the MAPS test set. The scenario is the same, as the validation set consists of pieces rendered by an unseen synthesizer. We feel that this also lends some additional strength to our argument, as we conduct our analysis on the best performing model for this set.

Two different scenarios are considered. The first scenario looks at the transcription results for notes and note combinations that are present in both the train and validation set, referred to as “shared” combinations. A low proportion of additions will tell us that there were a sufficient number of examples for this particular combination, so it could not be overshadowed by combinations containing additional notes. A high proportion of omissions will indicate issues with generalization to different acoustic properties. If both proportions are high, this indicates that one or more notes in the combination have been mistaken for others.

The second scenario examines the transcription results for notes and note combinations that are present only in the validation set, referred to as “unshared”. If the proportion of exactly transcribed frames is high, the network must have learned to disentangle individual notes from different combinations shown to it, and be able to recognize these disentangled parts in new, unseen combinations. A high proportion of additions will mainly tell us that the network has failed to disentangle parts, but still tries to combine the ones it knows about. A high proportion of omissions points to either a failure to simultaneously disentangle and recombine, a failure to generalize to different acoustic properties, or more probably both.

In figure [3] we can see two things: the most common note combinations present in both train and
validation set are actually isolated notes, and the relative frequency of exactly transcribed notes is comparatively high. Unfortunately, we can also see that the proportion of frames in which additional notes were erroneously transcribed is much higher than we would prefer, pointing to both a lack of examples for these individual notes at train time and the failure to generalize from combinations. They all are confused with combinations every so often. The low proportion of omission errors for isolated notes indicate only mild difficulties to generalize to different acoustical properties.

Looking at figure 4 we can see the error behavior of the network for the most common note combinations that are only present in the validation set. We notice a large amount of omission errors - which also indicates a failure to generalize to unseen note combinations. A few combinations, such as \((G3, A3, C4, D4)\), stand out though as being transcribed with great accuracy. We could find no satisfactory explanation for this so far, other than the suspicion it has to do with their low polyphony.

If we compare the results of the \textit{ConvNet} (figure 3) and the \textit{AUNet} (figure 5) for the most common note combinations which are shared by the train and validation set, we can observe that the \textit{AUNet} achieves marginally better exact transcription results across the board. In some cases, the proportion of added notes is reduced, however this happens at the expense of a slightly increased amount of omitted note combinations. Likewise, the results for the \textit{ConvNet} (figure 4) and \textit{AUNet} transcriptions (figure 6) for the “unshared” case appear to be very similar, indicating a comparable error behavior across very different architectures.

Concluding this section we would like to emphasize that both architectures achieve the same (or even slightly exceed) framewise transcription results on the MAPS dataset as reported in \cite{7}, which currently defines the state of the art. In other words, it is unlikely that the problematic results reported above are due to the fact that we made poor hyperparameter choices.

5 Summary

We have experimentally shown that certain neural network architectures have difficulties \textit{disentangling} inputs which are superpositions or mixtures of indi-
The brute force approach to solve the disentanglement problem would be showing all possible combinations to the network. Unfortunately this solution is intractable, due to the large tonal range and maximum polyphony of certain instruments. Arguably this approach would also not necessarily force the networks to learn how to disentangle, as they could, in principle, simply memorize all combinations. Learning a different note detector for each note, as done in [9, 10] suffers from the same problems, if the combinations shown to each detector are not diverse enough. Depending on the expressiveness of the model class, “diverse enough” could easily mean “all combinations”.

A partial solution to this problem might involve a modification of the loss function for the network. An additional objective must specify the need to disentangle individual notes explicitly. The network needs to learn to decompose a (nonlinear) mixture of signals into its constituent parts - a task commonly known as “source separation”. Finding a formulation of a joint objective combining multi-label losses with a separation encouraging penalty that solves this disentanglement problem is the topic of ongoing research.

Acknowledgements

This work is supported by the European Research Council (ERC Grant Agreement 670035, project CON ESPRESSIONE). The Tesla K40 used for this research was donated by the NVIDIA Corporation.

References

[1] Emmanouil Benetos, Sebastian Ewert, and Tillman Weyde. Automatic transcription of pitched and unpitched sounds from polyphonic music. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Flo-
[2] Nancy Bertin, Roland Badeau, and Gaël Richard. Blind signal decompositions for automatic transcription of polyphonic music: NMF and K-SVD on the benchmark. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007, pages 65–68, 2007.

[3] Nancy Bertin, Roland Badeau, and Emmanuel Vincent. Fast bayesian nmf algorithms enforcing harmonicity and temporal continuity in polyphonic music transcription. In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA '09, New Paltz, NY, USA, October 18-21, 2009, pages 29–32, 2009.

[4] Arnaud Dessein, Arshia Cont, and Guillaume Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010, pages 489–494, 2010.

[5] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. IEEE Trans. Audio, Speech & Language Processing, 18(6):1643–1654, 2010.

[6] Graham Grindlay and Daniel P. W. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA '09, New Paltz, NY, USA, October 18-21, 2009, pages 53–56, 2009.

[7] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016, pages 475–481, 2016.

[8] Anis Khlf and Vidhyasaharan Sethu. An iterative multi range non-negative matrix factorization algorithm for polyphonic music transcription. In Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26–30, 2015, pages 330–335, 2015.

[9] Matija Marolt. A connectionist approach to automatic transcription of polyphonic piano music. IEEE Trans. Multimedia, 6(3):439–449, 2004.

[10] Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011, pages 175–180, 2011.

[11] Ken O’Hanlon and Mark D. Plumbley. Polyphonic piano transcription using non-negative matrix factorisation with group sparsity. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014, pages 3112–3116, 2014.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III, pages 234–241, 2015.

[13] Siddharth Sigtia, Emmanuel Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur S. d’Avila Garcez, and Simon Dixon. A hybrid recurrent neural network for music transcription. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015, pages 2061–2065, 2015.
[14] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(5):927–939, 2016.

[15] Paris Smaragdis and Judith C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, page 177–180. IEEE, 2003.

[16] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Trans. Audio, Speech & Language Processing*, 18(3):528–537, 2010.

[17] Felix Weninger, Christian Kirst, Björn W. Schuller, and Hans-Joachim Bungartz. A discriminative approach to polyphonic piano note transcription using supervised non-negative matrix factorization. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6–10, 2013.

**Appendix**

![Figure 7: The ConvNet and SmallConvNet architectures. White boxes denote convolutional layers, black thick lines stand for fully connected layers. Arrows show information flow, dashed lines indicate a max-pooling operation.](image)

| Layer Type | Output Dimensions | No. of Params |
|------------|-------------------|---------------|
| Input      | 1x5x229           |               |
| Conv (Id)  | 32x5x229          | 288           |
| BatchNorm  | 32x5x229          | 128           |
| Relu       | 32x5x229          |               |
| Conv (Id)  | 32x3x227          | 9216          |
| BatchNorm  | 32x3x227          | 128           |
| Relu       | 32x3x227          |               |
| MaxPool    | 32x3x113          |               |
| Dropout, p=0.25 | 32x3x113    |               |
| Conv (Id)  | 64x1x111          | 18432         |
| BatchNorm  | 64x1x111          | 256           |
| Relu       | 64x1x111          |               |
| MaxPool    | 64x1x55           |               |
| Dropout, p=0.25 | 64x1x55   |               |
| Dense (Id) | 512               | 1802240       |
| BatchNorm  | 512               | 2048          |
| Relu       | 512               |               |
| Dropout, p=0.5 | 512           |               |
| Dense (Sigmoid) | 88              | 45144         |

Table 1: The ConvNet Architecture
Figure 8: A schematic drawing of the \textit{AUNet} architecture. White boxes denote convolutional layers, their width corresponds to the number of convolutional kernels, their height corresponds to the size of the resulting feature maps. Arrows show information flow, dashed lines indicate either a max-pooling operation, if the line goes from a higher to a lower box, or an upscaling operation if the line goes from a lower to a higher box. Grey boxes next to white boxes denote a concatenation of feature maps. We made two adaptations to the original UNet architecture. The first is the use of upscaling operations instead of deconvolutions, and the second adaptation is the last layer having convolutions with a large kernel width in the frequency direction.

| Layer   | Output Dimensions | No. of Params |
|---------|-------------------|---------------|
| Input   | 1x5x229           |               |
| Conv (Id) | 8x5x229          | 72            |
| BatchNorm | 8x5x229          | 32            |
| Relu    | 8x5x229           |               |
| Conv (Id) | 8x3x227          | 576           |
| BatchNorm | 8x3x227          | 32            |
| Relu    | 8x3x227           |               |
| MaxPool | 8x3x113/1x12     |               |
| Dropout, p=0.25 | 8x3x113      | 576           |
| Conv (Id) | 8x1x111/1x12    | 32            |
| BatchNorm | 8x1x111          |               |
| Relu    | 8x1x111           |               |
| MaxPool | 8x1x55/1x2       |               |
| Dropout, p=0.25 | 8x1x55        | 192           |
| Conv (Id) | 8x1x53/1x12     | 32            |
| BatchNorm | 8x1x53           |               |
| Relu    | 8x1x53            |               |
| MaxPool | 8x1x26/1x2       |               |
| Dropout, p=0.25 | 8x1x26        | 16            |
| Dense (Id) | 16              | 3328          |
| BatchNorm | 16            | 64            |
| Relu    | 16               |               |
| Dropout, p=0.5  | 16              | 512           |
| Dense (Sigmoid) | 23          | 391           |

\[ \sum \text{3327} \]

Table 2: The \textit{SmallConvNet} Architecture

| Layer   | Output Dimensions | No. of Params |
|---------|-------------------|---------------|
| Input   | 1x256x256         |               |
| Conv (Id) | 32x256x256        | 288           |
| BatchNorm | 32x256x256      | 128           |
| Elu     | 32x256x256        |               |
| Conv (Id) | 32x256x256        | 9216          |
| BatchNorm | 32x256x256      | 128           |
| Elu     | 32x256x256        |               |
| MaxPool | 32x128x128/3x3   |               |
| Conv (Id) | 32x128x128/3x3   | 9216          |
| BatchNorm | 32x128x128      | 128           |
| Elu     | 32x128x128        |               |
| Conv (Id) | 32x128x128/3x3   | 9216          |
| BatchNorm | 32x128x128      | 128           |
| Elu     | 32x128x128        |               |
| MaxPool | 32x64x64/3x3     |               |
| Conv (Id) | 64x64x64/3x3     | 18432         |
| BatchNorm | 64x64x64        | 256           |
| Elu     | 64x64x64          |               |
| Conv (Id) | 64x64x64/3x3     | 36864         |
| BatchNorm | 64x64x64        | 256           |
| Elu     | 64x64x64          |               |
| MaxPool | 64x32x32/2x2     |               |
| Conv (Id) | 64x32x32/2x2     | 36864         |
| BatchNorm | 64x32x32        | 256           |
| Elu     | 64x32x32          |               |
| Conv (Id) | 64x32x32/2x2     | 36864         |
| BatchNorm | 64x32x32        | 256           |
| Elu     | 64x32x32          |               |
| Conv (Id) | 64x32x32/2x2     | 36864         |
| BatchNorm | 64x32x32        | 256           |
| Elu     | 64x32x32          |               |
| MaxPool | 64x16x16/2x2     |               |
| Conv (Id) | 128x16x16/2x2    | 73728         |
| BatchNorm | 128x16x16       | 512           |
| Elu     | 128x16x16         |               |
| Conv (Id) | 128x16x16/3x3    | 147456        |
| BatchNorm | 128x16x16       | 512           |
| Elu     | 128x16x16         |               |
| Upscale | 128x32x32         |               |
| Concat  | 128x32x32         |               |
| Conv (Id) | 128x32x32/20x3   | 221184        |
| BatchNorm | 128x32x32       | 512           |
| Elu     | 128x32x32         |               |
| Conv (Id) | 128x32x32/20x3   | 147456        |
| BatchNorm | 128x32x32       | 512           |
| Elu     | 128x32x32         |               |
| Upscale | 128x32x32         |               |
| Concat  | 128x32x32         |               |
| Conv (Id) | 64x32x32/4x4     | 110592        |
| BatchNorm | 64x32x32/4x4   | 256           |
| Elu     | 64x32x32          |               |
| Conv (Id) | 64x32x32/4x4     | 256           |
| BatchNorm | 64x32x32/4x4   | 30644         |
| Elu     | 64x32x32          |               |
| Upscale | 64x32x32          |               |
| Concat  | 64x32x32          |               |
| Conv (Id) | 64x32x32/20x3    | 27648         |
| BatchNorm | 64x32x32/20x3   | 128           |
| Elu     | 64x32x32          |               |
| Conv (Id) | 64x32x32/20x3    | 9216          |
| BatchNorm | 64x32x32/20x3   | 128           |
| Elu     | 64x32x32          |               |
| Upscale | 64x32x32          |               |
| Concat  | 64x32x32          |               |
| Conv (Id) | 32x256x256/50x3  | 18432         |
| BatchNorm | 32x256x256/50x3 | 128           |
| Elu     | 32x256x256        |               |
| Conv (Id) | 32x256x256/50x3  | 9216          |
| BatchNorm | 32x256x256/50x3 | 128           |
| Elu     | 32x256x256        |               |
| Conv (Id) | 32x256x256/50x3  | 9216          |
| BatchNorm | 32x256x256/50x3 | 128           |
| Elu     | 32x256x256        |               |
| Conv (Sigmoid) | 1x256x389/1x4  | 1313          |

\[ \sum 694673 \]

Table 3: The \textit{AUNet} Architecture