International Journal of Geo-Information

MDPI

Article

Identification of Salt Deposits on Seismic Images Using Deep Learning Method for Semantic Segmentation

Aleksandar Milosavljević

Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia; aleksandar.milosavljevic@elfak.ni.ac.rs; Tel.: +381-18-529-331

Received: 20 November 2019; Accepted: 30 December 2019; Published: 1 January 2020

Abstract: Several areas of Earth that are rich in oil and natural gas also have huge deposits of salt below the surface. Because of this connection, knowing precise locations of large salt deposits is extremely important to companies involved in oil and gas exploration. To locate salt bodies, professional seismic imaging is needed. These images are analyzed by human experts which leads to very subjective and highly variable renderings. To motivate automation and increase the accuracy of this process, TGS-NOPEC Geophysical Company (TGS) has sponsored a Kaggle competition that was held in the second half of 2018. The competition was very popular, gathering 3221 individuals and teams. Data for the competition included a training set of 4000 seismic image patches and corresponding segmentation masks. The test set contained 18,000 seismic image patches used for evaluation (all images are 101×101 pixels). Depth information of the sample location was also provided for every seismic image patch. The method presented in this paper is based on the author's participation and it relies on training a deep convolutional neural network (CNN) for semantic segmentation. The architecture of the proposed network is inspired by the U-Net model in combination with ResNet and DenseNet architectures. To better comprehend the properties of the proposed architecture, a series of experiments were conducted applying standardized approaches using the same training framework. The results showed that the proposed architecture is comparable and, in most cases, better than these segmentation models.

Keywords: deep learning; convolutional neural networks; semantic segmentation; seismic imaging; salt deposits

1. Introduction

Seismic imaging enables the visualization of underground structures and it is used in the discovery of hydrocarbon fuel reserves. It is based on the emitting of sound waves that reflect on underground structures and are detected on a surface using receiver devices called geophones (see Figure 1). The reflected sound signal is recorded for further processing that leads to a three-dimensional (3D) representation of underground rocks structure [1]. Seismic images show the boundaries of different types of rocks. Theoretically, the strength of the rejected signal is directly proportional to the difference in the physical properties of the rocks at the point of contact. This practically means that seismic images contain information about boundaries between rocky deposits, while they expose very little about the rocks themselves [2]. Seismic images are used in the exploration of hydrocarbon fuel reserves by helping detect potential reservoir rocks and that is why identification of salt deposits plays an important role. The development of salt domes (see Figure 2) can deform surrounding rocks forming traps that hold oil and natural gas. Salt sediments have characteristics that make them both simple and difficult for identification. Salt density is usually about 2.14 g/cm^3, which is less than most nearby rocks.

Salt seismic velocity of around 4.5 km/s is usually bigger than the surrounding rocks. This difference causes a sharp reflection at the salt sediment boundary. Salt sediment is usually an amorphous form without some special internal structure, which means that there is typically not much reflection within the sediment itself unless it comes from some other rock that is trapped within [2].

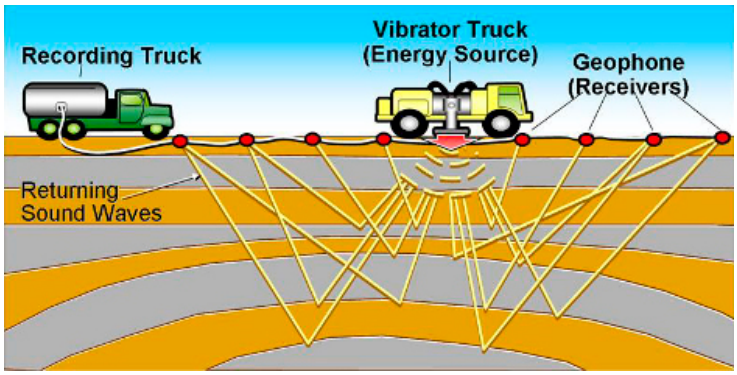

Figure 1. An illustration of the process of acquiring seismic images (source: www.petroman.co).

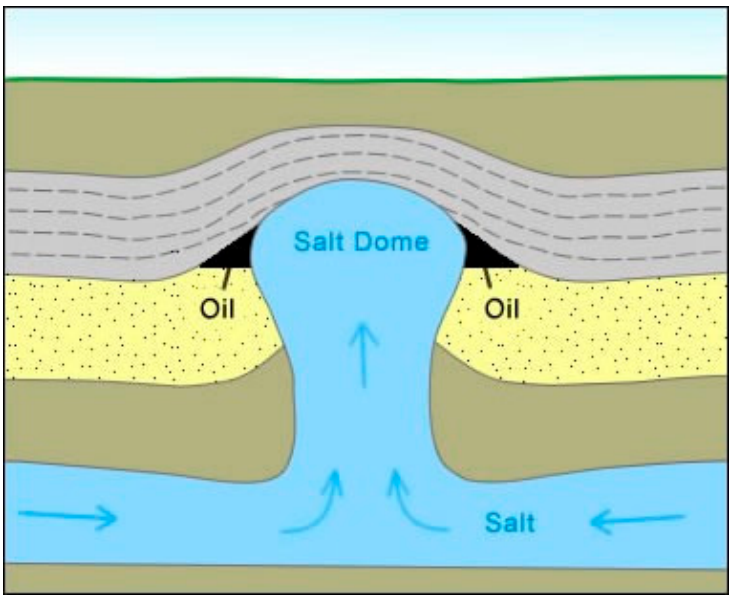

Figure 2. An illustration of a salt dome that is trapping oil (source: geology.com).

The goal of the TGS-NOPEC Geophysical Company (TGS) sponsored Kaggle competition [2] was to develop an algorithm that can classify each pixel in a 101×101 seismic image patch as salt or not salt. The data that was provided for training consists of 4000 seismic image patches along with appropriate binary masks that depict salt regions. For testing and scoring purposes, an additional 18,000 seismic image patches were provided. In addition, for each seismic image patch, depth information (in feet) is provided. Illustration of the dataset is depicted in Figure 3, where several pairs of seismic image patches and corresponding masks, along with depth information, are shown.

The way it is stated puts the problem into the semantic segmentation category. Semantic segmentation is one of the key problems in computer vision and represents a natural next step beyond image classification and object localization. Deep convolutional neural networks (CNN) [3] have been successfully applied to all these three problems. CNNs combine three architectural ideas that make them, to a certain degree, invariant to translation and distortion: local receptive fields, weights sharing, and spatial subsampling [4]. Deep CNNs are capable of building a hierarchy of features that makes them suitable for classification tasks, so as for the localization and semantic segmentation of objects in images. Although CNNs were known since the 1990s, deep CNNs are coming into the focus of

interest in 2012 when a network named AlexNet [5] won the ImageNet Large Scale Visual Recognition Challenge [6] (hereinafter referred to as ImageNet). AlexNet achieved a top-five error of 15.3% which was 10.8% better than the second-placed solution. This result was made possible by training using graphic processing units (GPU) which is considered a turning point for the progress of deep learning. In the following years, there has been a significant improvement in deep CNN architectures that lead to much better classification results in ImageNet challenge. In 2013, the winner was ZFNet [7] with a top-five error of 14.8%. The main contribution of the authors of this network is not that much in the classification result but in the development of visualization technique by mapping learned filters to image patches. The following year brought a huge advance and two significant solutions. VGGNet [8] got a top-five error of 7.3% while promoting simplicity and depth. The winner in 2014 was GoogLeNet [9] with a top-five of 6.7%. It is interesting to mention that GoogLeNet is not just much more accurate, but also has 12 times fewer parameters than AlexNet. In the next year, the error dropped to 3.6% which is almost twice as good result compared to the previous year. It was thanks to ResNet [10] architecture that is inspired by VGGNet simplicity with the introduction of residuals that enabled efficient training of deep CNNs. The 2015 winner was ResNet variation with 152 layers.

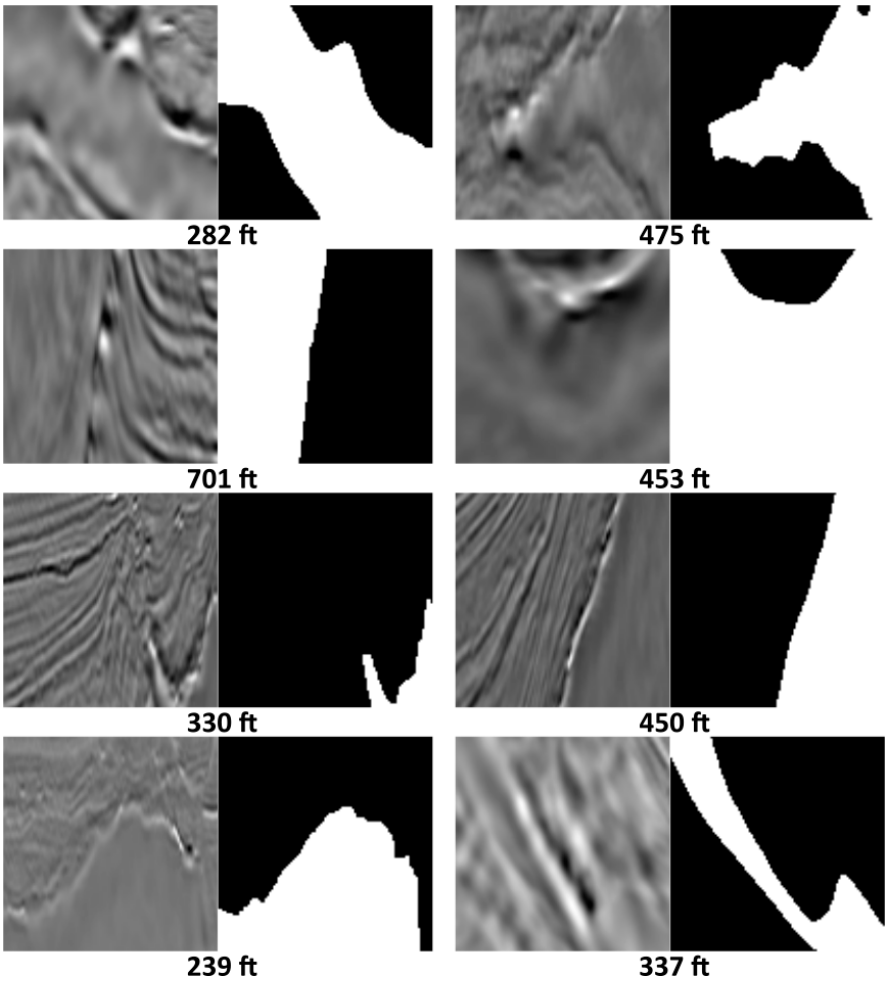

Figure 3. An example of several patches of the seismic image with corresponding salt deposit mask and depth information.

CNN architectures for classification are especially important because they can be relatively easily extended and used for semantic segmentation. There are many different architectures [11–18] that arise in the past couple of years to tackle this problem. In this paper, a novel architecture is proposed that came as a result of the author's participation in the TGS competition [2]. The architecture represents the

evolution of the U-Net [14] model integrating some ideas from ResNet [10] and DenseNet [19] architectures. Since the introduction of this architecture provided significant improvement of the competition score compared to vanilla U-Net, a series of post-competition experiments were conducted employing standardized segmentation models with the existing training framework. The results showed that the proposed architecture is comparable and, in most cases, better than tested segmentation models.

The paper is organized as follows: Section 2 presents an overview of related work concerning semantic segmentation and salt deposits identification. In Section 3, the proposed method is described presenting the used CNN architecture and some implementation details. Obtained results are presented and discussed in Section 4. Finally, Section 5 presents the conclusions.

2. Related Work

The problem of seismic image analyses and salt identification attracts many researchers. Typically for all computer vision problems, the traditional approach to seismic image analyses was based on hand-crafting different feature extractors and later processing of appropriate responses. In one of the first papers in this area, Pitas and Kotropoulos [20] suggested a method based on texture analyses for semantic segmentation of seismic images. Methods based on texture attributes remain feasible in recent years [21,22]. Harper and Clapp [23] suggested ways in which different seismic image attributes can be calculated and used to identify salt deposits. A method based on the calculation of new seismic image attributes is also used by Shafiq et al. [24]. Analysis of 3D seismic images was the research subject in [25–27]. Amin and Deriche [25] propose a method based on using a 3D multi-directional edge detector. Wu [26] relies on probability calculation to identify salt sediment boundaries. Di et al. [27] propose multi-attribute clustering using the *k*-means algorithm to identify salt sediment boundaries. The use of machine learning techniques to identify four characteristic seismic structures based on different seismic attributes extracted from the image was suggested by Wrona et al. [28].

Deep CNN architectures for classification, that arise in recent years, can be relatively easily extended and used for semantic segmentation. The fully convolutional network (FCN) [11] represents one of the first solutions following that path. An FCN is constructed by removing the top classifier layers of CNN, adding a convolutional layer to reduce the number of filters to the number of output classes, and upsampling the obtained feature map to match the input size of the network. Since the spatial component of the last CNN feature map is several times smaller than input size, to achieve better results it is possible to form a hypercolumn [12] using feature maps obtained from several stages of the CNN. The number of filters in each feature map is equalized with additional convolutional layers and, after upsampling to a certain size, an addition operation is used to construct a hypercolumn that is used for pixel-level classification.

More complex CNN architectures for semantic segmentation are DeconvNet [13] and U-Net [14]. Both networks share a similar architectural idea: the network consists of encoder and decoder parts. In the case of DeconvNet, they are called convolution and deconvolution networks respectively. The encoder part is basically a classification CNN with a removed top, while the decoder has a mirrored structure with blocks that consist of upsampling or transpose convolution layers followed by convolutional layers. U-Net architecture, in addition, introduces skip connections that enable copying and concatenating earlier encoder feature maps to corresponding upsampled decoder feature maps, so decoder convolution layers process them both. A very similar approach, with slight differences, is promoted by SegNet [15] architecture.

Deep learning models for semantic segmentation become increasingly popular in the automatic processing of remote sensing images for building footprint extraction [29–31], road extraction [32], land use detection [33], etc. Motivated by good results in many areas, CNNs are becoming researchers' default choice for the segmentation of seismic images and identification of salt deposits. A huge number of papers [34–41] in 2018 and 2019 supports the claim. Dramsch and Lüthje [34] evaluated several classification deep CNNs with transfer learning to identify nine different seismic textures

from 65 × 65 pixel patches. Di et al. [35] addressed the same problem using a deconvolutional neural network with three convolutional and three deconvolutional layers to speedup seismic interpretation. The same authors [36] addressed a problem of salt body delineation using a classification approach. They relied on the CNN with two convolutional and two fully connected layers, that was fed with 32 × 32 image patches. Waldeland et al. [37] proposed a method for the identification of salt bodies from 3D seismic data by classifying 65 × 65 × 65 data cubes using a custom CNN. Zeng et al. [38] proposed a method for salt body identification using the U-Net segmentation network in combination with the ResNet classification network to delineate salt bodies with high precision. Shi et al. [39] approached the problem of salt body extraction as 3D image segmentation and proposed an efficient approach based on deep CNN with an encoder–decoder architecture. They fed a CNN with 128 × 128 patches and predicted a per-pixel salt mask. Wu et al. [40] extended this approach to 3D and directly processed 128 × 128 × 128 data cubes to predict salt bodies. Finally, a paper by Babakhin et al. [41] should be emphasized since it describes the first-place solution on the TGS's challenge [2]. Their key 'ingredient to success' was a semi-supervised method which utilizes unlabeled (test) data for multi-round self-training. In the first round of training, they used only available labeled data, after that, in following rounds they trained the model on the available labeled data and predicted confident pseudo-labels for the unlabeled data. The architecture they employed used ResNet-34 [10] and ResNeXt-50 [42] encoders with scSE modules [43], feature pyramid attention (FPA) module [44] after the last encoder block, and hypercolumns [12] for predicting the segmentation mask.

The approach proposed in this paper uses segmentation CNN, unlike classification approaches presented in [34,36]. The prediction is performed on 2D images instead of approaches [37,40] that process 3D data cubes. It is based on a novel network architecture, that is presented in the next section, which distinguishes it from [35,38,39,41].

3. Method Description

This section presents the proposed network architecture that came out as a result of the author's participation in TGS sponsored Kaggle competition. Technical details regarding the implementation of the network and the training procedure that was applied are also included in this section.

3.1. Network Architecture

The architecture of the network that produced the final, i.e., the best solution provided by the author's participation in TGS's competition, is shown in Figure 4. The presented architecture originates from U-Net [14] and it includes certain modifications made in a search to improve the score.

As can be seen from Figure 4, the network mainly consists of three types of blocks (C, D, and U). C-block is the most common and the most complex block in the network. It consists of 5 convolutional layers with a kernel size of 3 × 3. The block is generated using two parameters: input number of filters (f), and an output number of filters (p). The first 4 layers have the same number of filters (f) and they are 'closely coupled' using addition before ReLU activation. The final, fifth, layer has p filters and its main use is to adjust the number of output filters to the desired value. The specific coupling of the layers' output maps was inspired by ResNet [10] and DenseNet [19] architectures, although the particular 'wiring' is the original contribution. In addition to previously mentioned convolutional layers, C-block also includes batch normalization and ReLU activation layers. The number of filters in convolutional layers of C-blocks is defined using parameter n that represents the number of filters in the first convolutional layers. The experiments were conducted with the following values for n: 16, 24, and 32.

D-block is the second block type that is used, and it can be found in the encoder part of the network always following C-block. The responsibility of D-block is the downsampling of the spatial dimension of the feature map by a factor of 2 using the MaxPool layer. It also includes the Dropout layer with a 20% dropout rate that is used to increase the robustness of the trained model and prevent overtraining. Dropout is only applied during the training phase and it represents annulment (setting

values to 0) of a certain percent (dropout rate) of the input feature map. This forces the network to take more input features into account when building higher-level features.

U-block is found in the decoder part of the network and it corresponds in a way to D-block in the encoder part. U-block is used for the upsampling spatial dimension of the input feature map by a factor of 2 which is achieved by duplicating each column and row. The second responsibility of U-block is a concatenation of a previously upsampled feature map with an output feature map from the appropriate C-block of the encoder (shown by the dashed line in Figure 4).

The final network output is obtained applying a 1×1 convolution, with 1 filter, and sigmoid activation to the output of the last C-block. The 1×1 convolution is used to reduce the number of filters to the desired network output, while sigmoid activation limits the output values per pixel to the interval (0, 1). The corresponding values are rounded to values 0 or 1 to produce the final output mask.

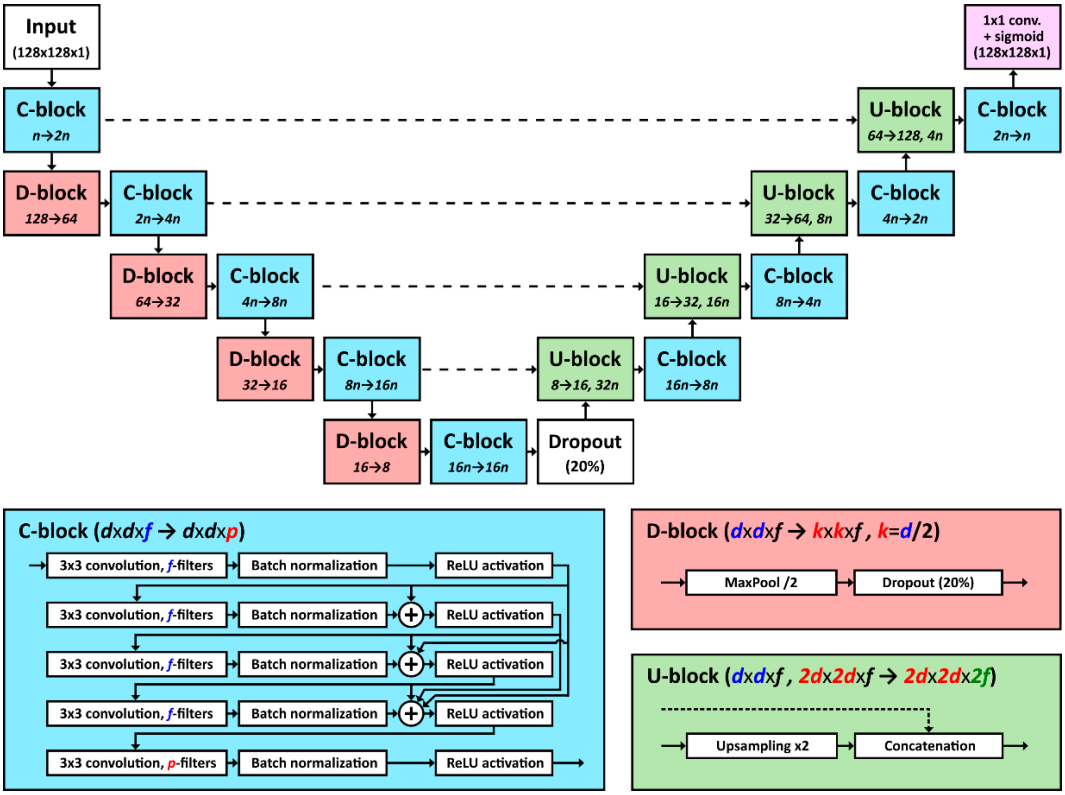

Figure 4. Overview of the architecture of the proposed convolutional neural network (CNN) for semantic segmentation of salt deposits. The network consists of three kinds of blocks (C, D, and U) that are showed in detail bellow. The final output is obtained by 1×1 convolution with 1 filter followed by the sigmoid activation function. The initial number of filters is indicated by n.

3.2. Implementation and Training

For implementing the proposed architecture, the Python programming language and the Keras [45] library were chosen. Keras is a high-level library that specifies a simplified interface for implementing deep neural networks and in this case, it relies on TensorFlow [46] as a backend engine. The particular network implementation uses Keras functional API, and each of the proposed architectural blocks is implemented in a separate Python function. The source code for the project can be found at https://github.com/a-milosavljevic/tgs-salt-identification.

The training was carried out on 80% of the initial training set, while the remaining 20% was used for validation. More precisely, the data were divided into 5 folds, so 5 networks were trained using the different fold for the validation. The final prediction was obtained using an ensemble of 5 networks were corresponding outputs were summed, divided by the number of the networks in the ensemble,

and then rounded to produce a binary mask. To keep validation sets representative compared to the training set, before splitting data into 5 folds, samples were sorted by the number of salt pixels in the mask. After that, images were split by taking 5 samples at a time and assigning them to each of the 5 allocated folds.

The data augmentation technique was applied in both training and test phase (results preparation phase). In the training phase, data augmentation was applied on two levels. The first level represents a doubling of the number of training samples by horizontal flipping all the images and corresponding masks. The second level of data augmentation used random image transforms that included translation, and intensity scaling and shifting. The translation transform relied on the fact that input images are 101×101, while the network input is 128×128. In 25% of the cases, the image was centered, i.e., it had a translation of 13 pixels in both directions. In the remaining 75% of the cases, the image was translated by random values between 0 and 27 in both directions. The appropriate translation was also applied to the corresponding mask. Besides translation, the unfilled part of the network input (and expected output) was filled by image and mask mirroring. Transformations related to image values include intensity scaling by multiplying image values with a random value between 0.8–1.2, such as intensity shifting by adding random value from the range ±0.2.

Test time augmentation (TTA) works as follows. For each image in the test set network output is evaluated for original and horizontally flipped variation. In the second case, the output is also flipped and averaged with the original output. When not only one, but the ensemble of networks, is used, averaging is done for all network outputs before thresholding operation that produces the final segmentation mask.

For the network training, *Adam* [47] optimizer (module *optimizers*) and *binary_crossentropy* loss function (module *losses*) were used. As the metric to select the optimal model *binary_accuracy* function from the *metrics* module was applied. This metric shows the percentage of correctly classified pixels. Also, the intersection over union (IoU) metric has been tracked and is calculated by dividing the number of pixels in the intersection of ground truth and predicted masks, and the number of pixels in the union of two masks. The IoU metric is typically used for segmentation problems, and the competition metric was based on it, but slightly better results were achieved using binary accuracy for selecting the optimal model.

To control training processes, Keras' callback objects were applied. Appropriate classes are part of the *callbacks* module and the proposed solution used: *ReduceLROnPlateau*, *EarlyStopping*, *ModelCheckpoint*, and *CSVLogger*. *ReduceLROnPlateau* is used to reduce learning rate using a certain factor when in a certain number of epochs certain parameter that is being tracked is not improved. In the proposed solution, binary accuracy on the validation set was tracked, and if there were no improvements after 10 epochs of training, the learning rate was reduced by 0.1. The initial value of the learning rate was 10-3. *EarlyStopping* callback, as the name implies, were used to end training using a similar mechanism to the previously described one. In the proposed solution, early stopping was activated after 30 epochs when binary accuracy on the validation set does not improve. *CSVLogger* callback was used to record training and validation losses and accuracies during the training process. Finally, *ModelCheckpoint* is used to record the best model measured using binary accuracy on the validation set. Each time the accuracy improves a model gets saved, so when the training ends, the model that had the best performance on the validation set is obtained.

4. Results and Discussion

All Kaggle hosted competitions have clearly defined duration, rules, datasets, and a metric for submission evaluation. In the case of TGS's competition training dataset consists of 4000 seismic image patches and corresponding segmentation masks that are 101×101 pixels in size. The test set that is used for model evaluation consists of 18,000 seismic image patches. For all seismic image patches, depth information (in feet) of the sample location is provided. The competition results consist of masks

that are estimated for every test image patch. The binary segmentation masks are encoded using the run-length encoding (RLE) and submitted using the standard comma-separated values (CSV) file.

The competition lasted from July 19 to October 20, 2018. Each day it was possible to submit a maximum of five solutions that are scored using a competition-specified metric. The preliminary (public) score is given based on 34% of the test data. The best result of a contestant determines its placement on the public leaderboard. Final results are announced after the competition ends and they are based on the remaining 66% of the test data. The final score is calculated for only two solutions selected by a contestant and the better value is used to place the contestant in a final (private) leaderboard. One week before the competition ends it is possible to merge teams which led to the merging of their individual results.

The main advantage of a competition organized in such a manner is the ability to try out and get an immediate assessment for many ideas in a short time, as well as to measure the quality of one's own solution compared to the other participants with a high level of objectivity.

4.1. Competition Results

The results achieved with the proposed network were ranked in the top 14%, i.e., as 446th of 3221 contestants. Before presenting the results, it is necessary to introduce the metric used by the competition to evaluate submissions. The metric was based on averaging IoU values on 10 thresholds ranging from 0.5 to 0.95 with a step of 0.05. For example, on threshold 0.5, the predicted mask is counted as a hit if the IoU value is greater than 0.5.

If T represents ground truth mask, while Y represents predicted mask, on each threshold value t precision $P(t)$ is calculated using the following rule:

$$P(t) = \begin{cases} 0, & |T| = 0 \wedge |Y| > 0 \\ 0, & |T| > 0 \wedge |Y| = 0 \\ 1, & |T| = 0 \wedge |Y| = 0 \\ IoU(T, Y) > t, & |T| > 0 \wedge |Y| > 0 \end{cases}. \quad (1)$$

Finally, evaluation metric M of the predicted mask is calculated by averaging precisions calculated at all 10 thresholds:

$$M = \frac{1}{10} \sum_{k=0}^{9} P(0.5 + 0.05k). \quad (2)$$

An overview of the results for different configurations regarding the number of networks in the ensemble and the number of filters in the first convolutional layer (parameter n) are presented in Table 1. Along with private scores that were the basis for the final ranking, corresponding public scores, that were visible during the competition, are also shown. For comparison purposes, the winning solution [41] score is also shown. The total number of submitted solutions for the winning team was 316, while the author had 42 total submissions.

Table 1. An overview of the results for different configurations.

Ensemble Size	Number of Filters in the First Layer (n)/Number of Networks	Public Score	Private Score
5	16/5	0.82519	0.84525
5	24/5	0.82859	0.84771
5	32/5	0.83181	0.85087
10	24/5, 32/5	0.83314	0.85202
25	16/15, 24/5, 32/5	0.83328	0.85241
	The result of the winning solution [41]	*0.88832*	*0.89646*
	The result of the first submitted solution	*0.74828*	*0.76996*

In the presented results two trends can be spotted. The first trend is score increase with the number of convolutional filters (parameter n), while the second one suggests that adding more networks to an ensemble also pays off. The best author's submission used 25 networks which may sound surprising. It was a last-minute attempt to increase the score by combining all previously trained models. As a comparison, the winning solution came from the ensemble of 40 networks.

The things that are not visible from the presented results are countless attempts that did not prove successful. These attempts include other architectures, different loss functions and metrics for picking the best model, some postprocessing efforts involving morphology operations, other methods for data augmentation, etc.

As an illustration of training processes, Figure 5 depicts two charts showing the accuracy and IoU metrics on the training and validation sets during one round of training.

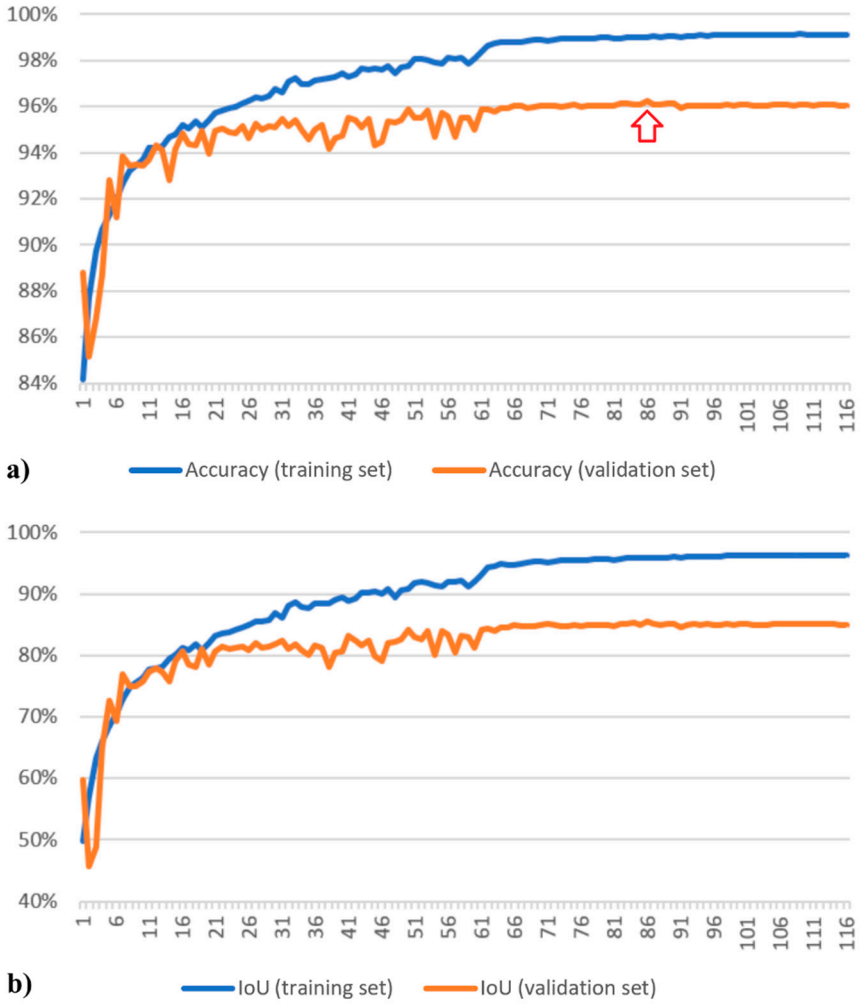

Figure 5. A sample accuracy (**a**) and intersection over union (IoU) (**b**) metrics charts showing values on the training and validation sets. The data are taken from the training of the fifth model with 32 initial filters. The red arrow in (**a**) depicts the best accuracy epoch when the model is saved.

4.2. Post-Competition Analyses

Judging the model architecture just by comparing the competition results is not the best approach since many other things influence the final score. For that reason, after the competition ended, a few experiments were conducted to see how different segmentation models, using existing training framework, would rank compared to the proposed model. Luckily, thanks to Pavel Yakubovskiy's Github library called Segmentation Models [48], that was fairly easy. Segmentation models are

a Keras-based Python library that implements four popular segmentation architectures and dozens of ImageNet pretrained backbones that can easily be combined into the segmentation model of choice. The supported architectures include previously mentioned U-Net [14], such as the feature pyramid network (FPN) [16], LinkNet [17], and pyramid scene parsing network (PSPNet) [18].

When it comes to a backbone selection, the segmentation models library supports a range of models pretrained on the 2012 ILSVRC ImageNet dataset. For the experiments, ResNet-34 [10] backbone is used. Since ResNet network decreases the size of feature maps four times in first two layers, to preserve the precision of predicted output map, input images were upscaled two times by adding *UpSampling2D* layer at the input and adjust output size by applying *AvgPool2D* on the output (see Figure 6 for the illustration). A similar method of doubling the size of input images was applied in the winning solution, as described in [41].

A comparison of different segmentation models is presented in Table 2. For each of the models, a total number of convolutional layers and filters—public and private scores are shown. For scoring purposes, ensembles of five networks were used where each of the networks was trained using the different fold as the validation set. The training process, including data augmentation during both training and results preparation phase, was identical to the one used in the competition and previously described in detail in Section 3.2. The first two entries correspond to the results obtained using the network described in this paper. The first entry represents the result obtained during the competition, while the second one was produced by a slightly modified network that excludes dropout layers in D-blocks. Finally, the last four entries correspond to the models implemented using Yakubovskiy's library. FPN, LinkNet, and PSPNet were constructed using default parameters, while the U-Net model was constructed with 512, 256, 128, 64, and 32 filters in each of the decoder blocks which is double the default. This change was introduced to give U-Net similar 'power', in terms of the number of convolutional filters, to the one the original model had.

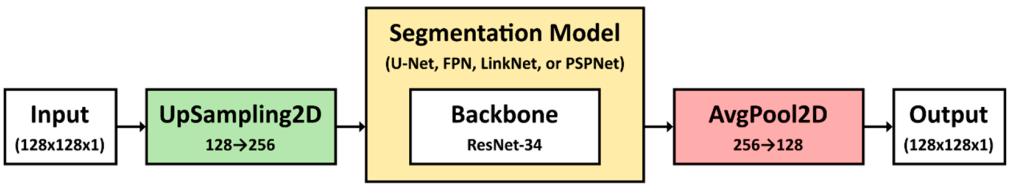

Figure 6. An illustration of the experiment framework based on Yakubovskiy's segmentation models [48] and input image upscaling. Please note that in the case of PSPNet, due to network limitations, an input size of 120 x 120 (upscaled to 240 x 240) was used.

Table 2. A post-completion comparison of different segmentation models.

Segmentation Model	Number of Convolutional Layers/Filters	Public Score	Private Score
Original model (n = 32)	46/9761	0.83181	0.85087
Original model w/o dropout	46/9761	0.83108	**0.85219**
U-Net (n = 32)	48/10561	0.82446	0.84404
FPN	51/11137	**0.83623**	0.85145
LinkNet	53/9617	0.82591	0.84565
PSPNet	23/4225	0.83137	0.81138

Looking at the scores presented in Table 2 it can be noticed that the values do not differ much. The best public score is achieved by the FPN model while the winner in the private score section was the proposed model without dropout layers. Interestingly, looking at public scores, the unchanged original model had a slightly better score. The results also showed that original U-Net architecture proves to be less powerful than the proposed architecture, even though it used a pretrained ResNet-34 encoder as the backbone. The FPN model showed comparable—and even the best result in the public section—employing slightly more convolutional filters that both the proposed and U-Net models.

Finally, LinkNet and PSPNet performed slightly worse than the proposed model, but it is worth mentioning that PSPNet uses about two times fewer convolutional filters than the other models.

To further explore how these segmentation models behave, the outputs were visualized and compared. Figure 7 depicts six sample images along with ground truth salt deposits mask and output masks for each of the models. The examples were handpicked to illustrate some easy and some hard cases. The samples were chosen from the validation sets, so the outputs are produced by the networks that have not seen these inputs during the training.

The first three samples can be treated as the easy ones, i.e., all models' outputs show very good matching with the ground truth image. The fourth sample is slightly more complex, so differentiation between the models can be observed, however mostly depicting the 'right shape'. The last two samples are representatives of hard cases where the true mismatch occurs. The ground truth images for both samples show no presence of salt deposits, but the output images have plenty of it. Finding a way to reduce the occurrence of such cases seems like a good research direction.

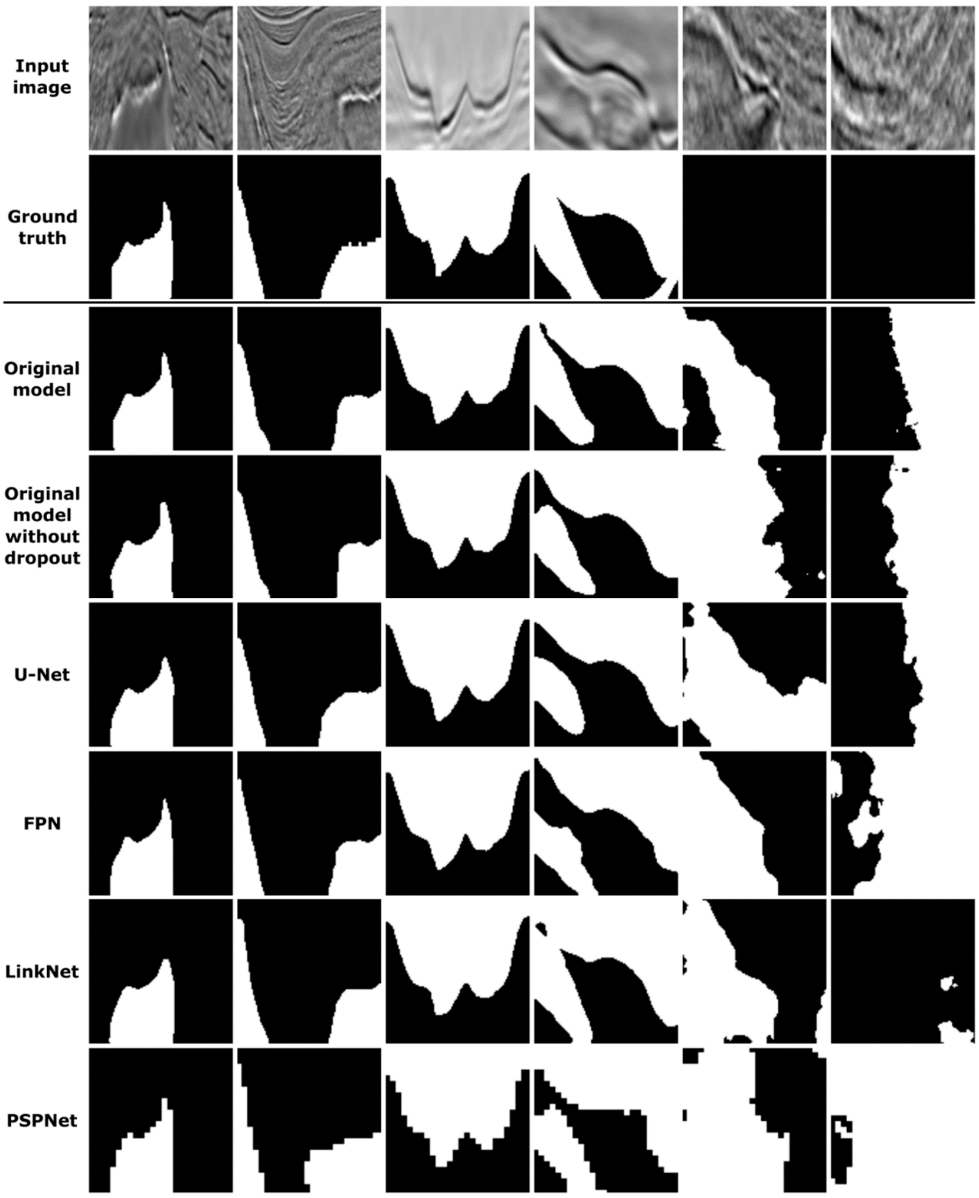

Figure 7. An example of several input seismic images with corresponding salt deposit masks (ground truth) along with predicted masks from all the models involved in the post-competition analyses. The samples are selected to illustrate both easy and hard cases.

For the comparison with the original model training charts depict in Figure 5, corresponding accuracy and IoU metrics charts are presented in Figures 8–12. Figure 8 depicts charts for the original model without dropout, while Figures 9–12 depicts charts for Yakubovskiy's segmentation models U-Net, FPN, LinkNet, and PSPNet, respectively. Again, the chart data are taken from the training of the fifth model (last partition) and the red line in (a) depicts the best accuracy epoch when the model is saved. Since the same training framework is used, all these charts show similar behavior. The oscillations of the validation metrics are stronger initially, and then stabilizes when the learning rate decreases. It appears that the oscillations are smallest for the original model (Figure 5) and that is probably due to the dropout that has been applied. If we compare the IoU metric for the training set, it can be noticed that, after the first epoch of training, our models get ~50%, while Yakubovskiy's models go to ~60%. This behavior can be explained by the fact that Yakubovskiy's models use a pretrained ResNet-34 encoder that gives them an initial boost.

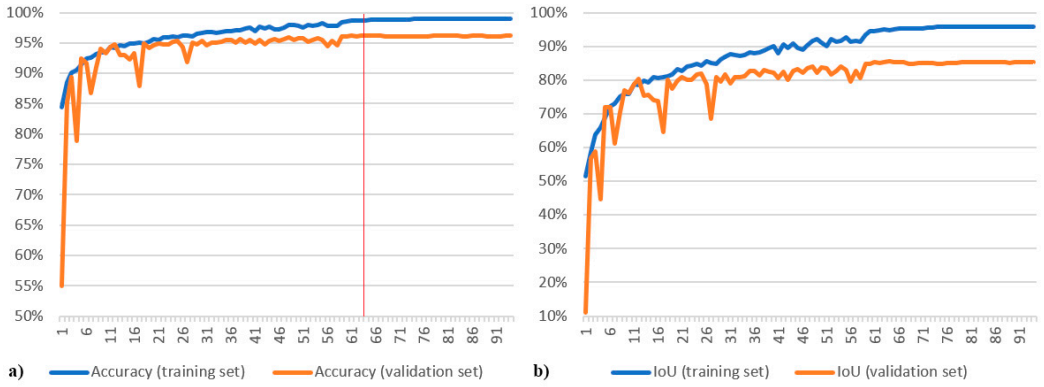

Figure 8. Accuracy (**a**) and IoU (**b**) metrics for the original model without dropout (n = 32).

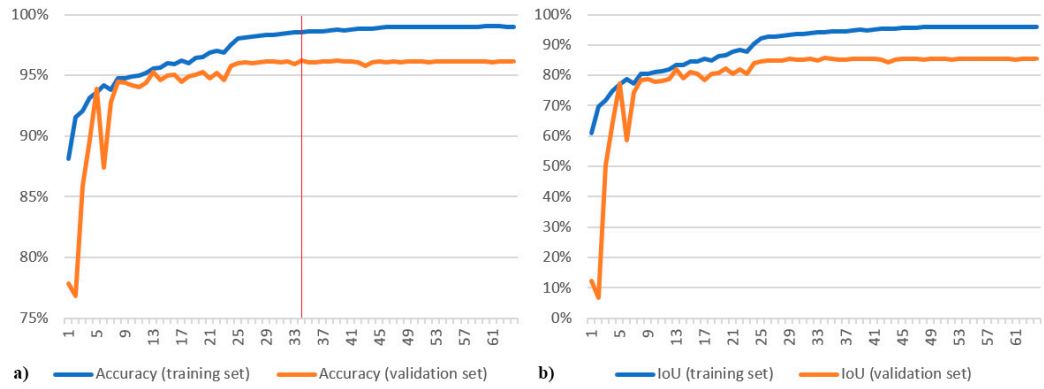

Figure 9. Accuracy (**a**) and IoU (**b**) metrics for the U-Net (n = 32) model.

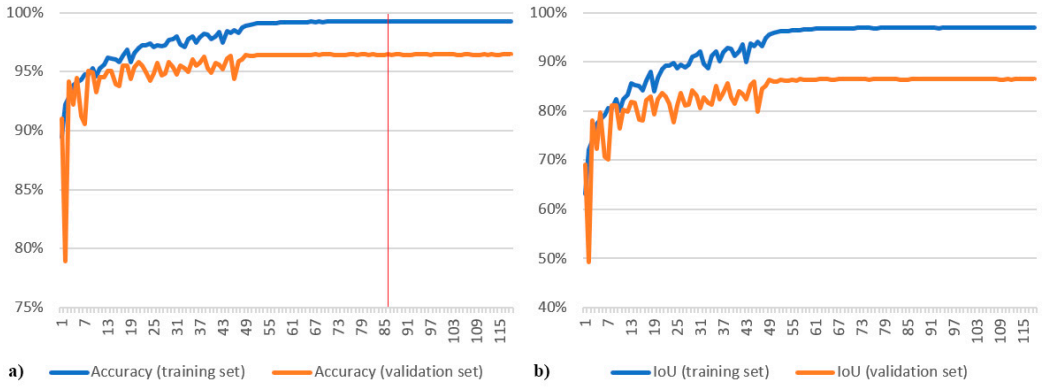

Figure 10. Accuracy (**a**) and IoU (**b**) metrics for the feature pyramid network (FPN) model.

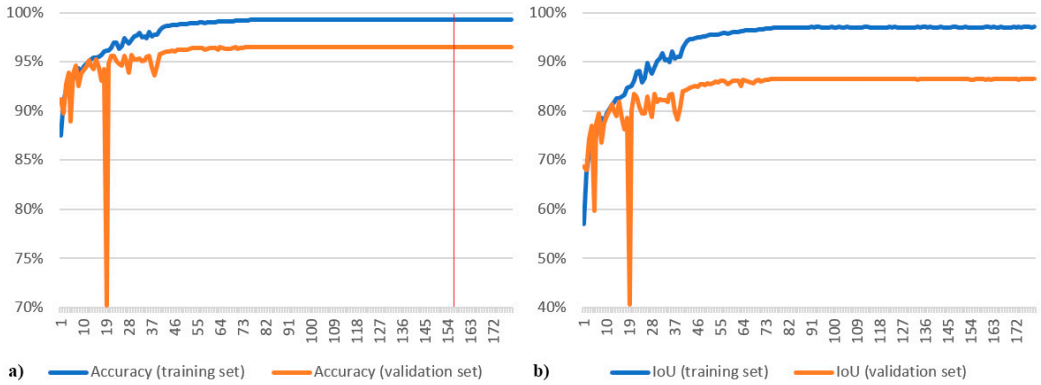

Figure 11. Accuracy (**a**) and IoU (**b**) metrics for the LinkNet model.

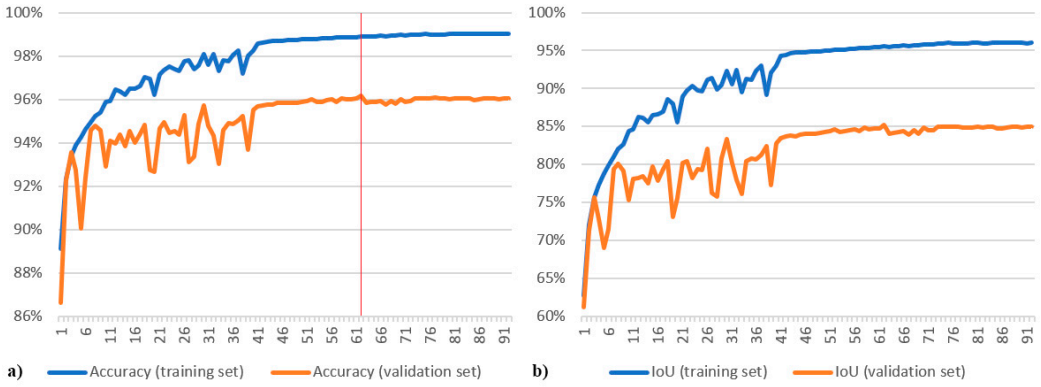

Figure 12. Accuracy (**a**) and IoU (**b**) metrics for the PSPNet model.

5. Conclusions

In this paper, an approach for the identification of underground salt deposits on seismic images using deep CNN for segmentation is presented. Seismic imagining and salt identification play a significant role in oil and gas discovery, while automatization of the process of their analyses is highly important for the companies involved in hydrocarbon fuel exploration.

This paper came as a result of the author's participation in the TGS sponsored Kaggle competition [2]. The solution that has a score of 0.85241 on the private leaderboard was based on the original architecture of segmentation CNN that was described in the paper. The proposed architecture was derived from the U-Net model that was responsible for the author's initial score of 0.76996. Since the improvement was significant, to better comprehend the properties of the proposed architecture, a series of post-competition experiments were conducted to compare obtained results with the results achieved by standardized approaches using the same training framework. The results show that the proposed approach is comparable and, in most cases, better than these segmentation models. The five models' ensemble of the proposed architecture without dropout layers had the best private score of 0.85219, while the FPN-based ensemble recorded the best public score of 0.83623. It is worth mentioning that segmentation models used in the comparison employed transfer learning, i.e., they used ResNet-34 encoders pretrained on the ImageNet 2012 dataset.

For future work, several directions can be examined. The first one is to apply transfer learning to the proposed architecture by replacing the current encoder with some pretrained network and only keep the decoder part. The second direction would be to try to improve salt identification metrics by training additional classification models that would predict if a patch has salt or not, so the segmentation model would be used conditionally. Finally, the third direction would be to test the architecture on some other dataset and see if it holds up.

Author Contributions: Conceptualization, methodology, validation, formal analysis, supervision, writing—review and editing, data curation, Aleksandar Milosavljević. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This study was supported by the Ministry of Education, Science and Technological Development, Republic of Serbia, as part of the projects III-43007 and III-47003.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chevron.com: Seismic Imagining. Available online: https://www.chevron.com/stories/seismic-imaging (accessed on 17 November 2019).
2. Kaggle.com: TGS Salt Identification Challenge, Segment Salt Deposits Beneath the Earth's Surface. Available online: https://www.kaggle.com/c/tgs-salt-identification-challenge (accessed on 17 November 2019).
3. LeCun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; Jackel, L. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2*; The MIT Press: Cambridge, MA, USA, 1990; pp. 396–404.
4. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw.* **1995**, *3361*, 1995.
5. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, Spain, 3–8 December 2012; Volume 2, pp. 1097–1105.
6. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
7. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; LNCS; Springer: Cham, Switzerland, 2014; Volume 8689, pp. 818–833.
8. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015; ICLR: London, UK, 2015.
9. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–9.
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 770–778.
11. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; CVPR: Washington, WA, USA, 2015; pp. 3431–3440.
12. Hariharan, B.; Arbeláez, P.; Girshick, R.; Malik, J. Hypercolumns for object segmentation and fine-grained localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; CVPR: Washington, WA, USA, 2015; pp. 447–456.
13. Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Boston, MA, USA, 7–12 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1520–1528.
14. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2015; Volume 9351, pp. 234–241.
15. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]

16. Kirillov, A.; He, K.; Girshick, R.; Dollár, P. A Unified Architecture for Instance and Semantic Segmentation. Available online: http://presentations.cocodataset.org/COCO17-Stuff-FAIR.pdf (accessed on 19 November 2019).
17. Chaurasia, A.; Culurciello, E. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In Proceedings of the 2017 IEEE Visual Communications and Image Processing, VCIP 2017, St. Petersburg, FL, USA, 10–13 December 2017; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2018; pp. 1–4.
18. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2881–2890.
19. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2017; pp. 2261–2269.
20. Pitas, I.; Kotropoulos, C. A texture-based approach to the segmentation of seismic images. *Pattern Recognit.* **1992**, *25*, 929–945. [CrossRef]
21. Hegazy, T.; AlRegib, G. Texture attributes for detecting salt bodies in seismic data. In Proceedings of the Society of Exploration Geophysicists International Exposition and 84th Annual Meeting SEG 2014, Denver, CO, USA, 26–31 October 2014; Society of Exploration Geophysicists: Tulsa, OK, USA, 2014; pp. 405–408.
22. Shafiq, M.A.; Wang, Z.; Amin, A.; Hegazy, T.; Deriche, M.; AlRegib, G. Detection of salt-dome boundary surfaces in migrated seismic volumes using gradient of textures. In *SEG Technical Program Expanded Abstracts*; Society of Exploration Geophysicists: Tulsa, OK, USA, 2015; Volume 34, pp. 1811–1815.
23. Halpert, A.; Clapp, R.G. Salt Body Segmentation with Dip and Frequency Attributes, Stanford Exploration Project, Report SEP134. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.483.6634&rep=rep1&type=pdf#page=119 (accessed on 19 November 2019).
24. Shafiq, M.A.; Alshawi, T.; Long, Z.; Alregib, G. SalSi: A new seismic attribute for salt dome detection. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, Shanghai, China, 20–25 March 2016; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2016; pp. 1876–1880.
25. Asjad, A.; Mohamed, D. A new approach for salt dome detection using a 3D multidirectional edge detector. *Appl. Geophys.* **2015**, *12*, 334–342. [CrossRef]
26. Wu, X. Methods to compute salt likelihoods and extract salt boundaries from 3D seismic images. *Geophysics* **2016**, *81*, IM119–IM126. [CrossRef]
27. Di, H.; Shafiq, M.; AlRegib, G. Multi-attribute k -means clustering for salt-boundary delineation from three-dimensional seismic data. *Geophys. J. Int.* **2018**, *215*, 1999–2007. [CrossRef]
28. Wrona, T.; Pan, I.; Gawthorpe, R.L.; Fossen, H. Seismic facies analysis using machine learning. *Geophysics* **2018**, *83*, O83–O95. [CrossRef]
29. Schuegraf, P.; Bittner, K. Automatic Building Footprint Extraction from Multi-Resolution Remote Sensing Images Using a Hybrid FCN. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 191. [CrossRef]
30. Liu, H.; Luo, J.; Huang, B.; Hu, X.; Sun, Y.; Yang, Y.; Xu, N.; Zhou, N. DE-Net: Deep Encoding Network for Building Extraction from High-Resolution Remote Sensing Imagery. *Remote Sens.* **2019**, *11*, 2380. [CrossRef]
31. Alidoost, F.; Arefi, H.; Tombari, F. 2D Image-To-3D Model: Knowledge-Based 3D Building Reconstruction (3DBR) Using Single Aerial Images and Convolutional Neural Networks (CNNs). *Remote Sens.* **2019**, *11*, 2219. [CrossRef]
32. Wu, S.; Du, C.; Chen, H.; Xu, Y.; Guo, N.; Jing, N. Road Extraction from Very High Resolution Images Using Weakly labeled OpenStreetMap Centerline. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 478. [CrossRef]
33. Li, L. Deep Residual Autoencoder with Multiscaling for Semantic Segmentation of Land-Use Images. *Remote Sens.* **2019**, *11*, 2142. [CrossRef]
34. Dramsch, J.S.; Lüthje, M. Deep learning seismic facies on state-of-the-art CNN architectures. In Proceedings of the 2018 SEG International Exposition and Annual Meeting, SEG 2018, Anaheim, CA, USA, 14–19 October 2018; Society of Exploration Geophysicists: Tulsa, OK, USA, 2019; pp. 2036–2040.

35. Di, H.; Wang, Z.; AlRegib, G. Real-time seismic image interpretation via deconvolutional neural network. In Proceedings of the 2018 SEG International Exposition and Annual Meeting, SEG 2018, Anaheim, CA, USA, 14–19 October 2018; Society of Exploration Geophysicists: Tulsa, OK, USA, 2019; pp. 2051–2055.
36. Di, H.; Wang, Z.; AlRegib, G. Deep Convolutional Neural Networks for Seismic Salt-Body Delineation. In Proceedings of the AAPG Annual Convention and Exhibition 2018, Utah, UT, USA, 20–23 May 2018.
37. Waldeland, A.U.; Jensen, A.C.; Gelius, L.-J.; Solberg, A.H.S. Convolutional neural networks for automated seismic interpretation. *Lead. Edge* **2018**, *37*, 529–537. [CrossRef]
38. Zeng, Y.; Jiang, K.; Chen, J. Automatic seismic salt interpretation with deep convolutional neural networks. In Proceedings of the ACM International Conference Proceeding Series, Cambridge, UK, 23–26 March 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 16–20.
39. Shi, Y.; Wu, X.; Fomel, S. SaltSeg: Automatic 3D salt segmentation using a deep convolutional neural network. *Interpretation* **2019**, *7*, SE113–SE122. [CrossRef]
40. Wu, X.; Liang, L.; Shi, Y.; Fomel, S. FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation. *Geophysics* **2019**, *84*, IM35–IM45. [CrossRef]
41. Babakhin, Y.; Sanakoyeu, A.; Kitamura, H. Semi-Supervised Segmentation of Salt Bodies in Seismic Images using an Ensemble of Convolutional Neural Networks. In *Pattern Recognition*; Springer: Cham, Switzerland, 2019.
42. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K.; San Diego, U. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1492–1500.
43. Roy, A.G.; Navab, N.; Wachinger, C. Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2018; Volume 11070 LNCS, pp. 421–429.
44. Li, H.; Xiong, P.; An, J.; Wang, L. Pyramid attention network for semantic segmentation. In Proceedings of the British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, 3–6 September 2018.
45. Keras: The Python Deep Learning Library. Available online: https://keras.io (accessed on 16 October 2019).
46. TensorFlow: An End-to-end Open Source Machine Learning Platform. Available online: https://www.tensorflow.org/ (accessed on 16 October 2019).
47. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
48. Yakubovskiy, P. Segmentation Models, Github Library. Available online: https://github.com/qubvel/segmentation_models (accessed on 19 November 2019).