Convolutional and Residual Networks Provably Contain Lottery Tickets

Rebekka Burkholz [1]

Abstract

The Lottery Ticket Hypothesis continues to have a profound practical impact on the quest for small scale deep neural networks that solve modern deep learning tasks at competitive performance. These lottery tickets are identified by pruning large randomly initialized neural networks with architectures that are as diverse as their applications. Yet, theoretical insights that attest their existence have been mostly focused on deep fully-connected feed forward networks with ReLU activation functions. We prove that also modern architectures consisting of convolutional and residual layers that can be equipped with almost arbitrary activation functions can contain lottery tickets with high probability.

1. Introduction

The Lottery ticket (LT) Hypothesis (Frankle & Carbin, 2019) has fueled the interest in deep neural network pruning to a reduce the number of trainable parameters with the purpose to save computational resources, regularize, and perform meaningful structure learning. Most newly developed algorithms are benchmarked and evaluated in the imaging domain. Naturally, most architectures that are pruned in practice contain therefore convolutional layers. In particular residual blocks, and skip connections in general, seem to provide iterative pruning algorithms an advantage over less computationally cumbersome approaches (Ma et al., 2021). It is subject of an ongoing debate to which degree different algorithms are successful in finding task specific computational neural network structures (Su et al., 2020; Ma et al., 2021; Fischer & Burkholz, 2022) and whether LTs are identifiable by contemporary pruning algorithms to solve complex problems with large scale architectures (Frankle et al., 2020; Renda et al., 2020). Theoretical insights into the conditions when we can expect to find LTs can provide guidance regarding when improvements could be feasible.

Our work contributes to the discussion with the assurance that LTs likely exist under realistic conditions in convolutional networks with or without residual blocks even when pruning algorithms are currently challenged to find them. This is in line with the Strong Lottery Ticket Hypothesis (SLTH), which has been posed by (Ramanujan et al., 2020) based on experiments in inspiration of (Zhou et al., 2019). It suggests that a sufficiently large neural network with random parameters contains, with high probability, for each target network (of certain maximal size) a sub-network that can approximate the target network with high accuracy. Such a sub-network is also called strong LT and does not need to be trained in order to achieve a performance that is competitive with the one of the target network.

The existence of such strong LTs has been proven for fully-connected feed forward architectures and ReLU activation functions by providing a probabilistic lower bound on the required width of the larger random network. First proven by (Malach et al., 2020), the width requirements have succinctly been improved to a logarithmic factor in the relevant variables (Pensia et al., 2020; Orseau et al., 2020) and extended to nonzero biases (Fischer & Burkholz, 2021). The only results for convolutional architectures are provided by (Burkholz et al., 2022; da Cunha et al., 2022) but either apply to specific targets (Burkholz et al., 2022) or are restricted to positive inputs (da Cunha et al., 2022) and do not cover common data transformations. Furthermore, all of these results rely on random networks that have at least twice the depth of the target network. This excludes constructions, in which the LTs can have residual blocks of the same size as in the target network, and reduces the expressiveness of the target networks, as they cannot utilize a large part of the available depth for a sparser representation (Yarotsky, 2018; Mhaskar et al., 2017). To overcome these limitations, we follow a similar strategy as (Burkholz, 2022) to extend existence results to a larger class of activation functions and random networks that have a similar depth as the target network ($L+1$). We solve the additional challenge to prune and construct convolutional filters with and without skip connections and residual blocks. In doing so, we propose a different construction than (da Cunha et al., 2022) that is not restricted to positive inputs.

*Equal contribution [1]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Rebekka Burkholz <burkholz@cispa.de>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

1.1. Contributions

1) We prove the existence of strong lottery tickets in convolutional neural network architectures, potentially with skip connections and residual blocks. 2) Our constructions are not restricted to CNNs with positive inputs and ReLUs in contrast to (da Cunha et al., 2022). 3) Our proofs apply to a large class of activation functions, including ReLUs, Leaky ReLUs, Tanh, and Sigmoids. 4) We present two types of constructions: (a) one in which the large, randomly initialized neural network that contains LTs has at least twice the depth of a target network, i.e. $L_0 = 2L_t$; and one in which the target can leverage almost the full depth of the large network for a sparser representation, as $L_0 = L_t + 1$. 5) We verify in experiments that our theory derives realistic conditions. Based on insights on solving subset sum approximation problems experimentally, we assess the expected sparsity of our LTs.

1.2. Related Literature

Most LT experiments are conducted in the context of image classification and thus rely heavily on pruning convolutional and residual neural network architectures to reduce the number of trainable parameters of a neural network (LeCun et al., 1990a; Mozer & Smolensky, 1989; Han et al., 2015; Frankle & Carbin, 2019; Srinivas & Babu, 2016; Lee et al., 2020; You et al., 2020; Frankle et al., 2020; Renda et al., 2020; Liu et al., 2021a; Liu et al.; Weigend et al., 1991; Savarese et al., 2020a; Chen et al., 2021c; Savarese et al., 2020b; LeCun et al., 1990b; Hassibi & Stork, 1992; Dong et al., 2017; Li et al., 2017; Molchanov et al., 2017; Zhang et al., 2021c). Some exceptions include graph neural networks (Chen et al., 2021b) and GANs (Chen et al., 2021a), which still utilize convolutions. One of the main objectives is to reduce the computational burden associated with deep learning. This can also be achieved with the help of core sets (Zhang et al., 2021b) or by starting the pruning not from a dense but a sparse random architecture (Evci et al., 2020; Liu et al., 2021b). Pruning before training (Wang et al., 2020; Lee et al., 2019; Verdenius et al., 2020; Tanaka et al., 2020; Ramanujan et al., 2020) is also a promising research direction but iterative pruning methods often perform better (Frankle et al., 2021; Ma et al., 2021; Fischer & Burkholz, 2022), while most benefits seem to result from residual skip connections (Ma et al., 2021). Another objective in the identification of LTs is structure learning, which seems to be more effective at lower sparsity levels (Su et al., 2020; Lee et al., 2020) and in many cases Iterative Magnitude Prunings (IMP) (Han et al., 2015; Frankle & Carbin, 2019) can fail to find structures that perform superior to random or smaller dense networks (Ma et al., 2021). Regardless, pruning can have provable regularization and generalization properties (Zhang et al., 2021a). At least for fully-connected architectures it has also been shown that structurally relevant LTs exist theoretically.

Most of the discussed pruning methods try to find weak LTs by identifying a sparse neural network architecture that is well trainable. Strong LTs are sparse sub-networks that perform well even without training and just rely on their initial parameters (Zhou et al., 2019; Ramanujan et al., 2020) and are thus also weak LTs. Their existence has been proven for fully-connected feed forward networks with RELU activation functions by providing lower bounds on the width of the large, randomly initialized neural network that contains them (Malach et al., 2020; Pensia et al., 2020; Orseau et al., 2020; Fischer & Burkholz, 2021; Burkholz et al., 2022). In addition, it was shown that multiple candidate tickets exist that are also robust to parameter quantization (Diffenderfer & Kailkhura, 2021). These works are restricted to ReLUs and always assume that the large randomly initialized neural networks has at least twice the depth of a target network $L_0 \geq 2L_t$. (Burkholz, 2022) extends these results to more general activation functions and also introduces a strategy to handle $L_0 \geq L_t + 1$. Up to our knowledge, (Burkholz et al., 2022; da Cunha et al., 2022) are the only theoretical works on convolutional architectures and apply only to ReLU activation functions. While (Burkholz et al., 2022) uses convolutions to obtain specific representations of basis functions, (da Cunha et al., 2022) studies general convolutional layers but is restricted to target networks with positive inputs, which does not cover the common image transformation procedures. We present more general results that also apply to potentially negative inputs, require less depth, can handle skip connections, which seem to be essential for the success of state-of-the-art pruning algorithms (Ma et al., 2021), and cover a large class of activation functions.

2. Background and notation

Let a convolutional neural network $f : \mathcal{D} \subset \mathbb{R}^{c_0 \times d_0} \to \mathbb{R}^{c_L \times d_L}$ be defined on a compact domain $\mathcal{D}$ and have channels $\bar{c} = [c_0, c_1, ..., c_L]$, i.e., depth L and width c_l in layer $l \in [L] := \{0, ..., L\}$. It is equipped with a continuous activation function $\phi(x)$ that has Lipschitz constant T on a compact domain that includes the possible inputs. f maps an input tensor $\boldsymbol{x}^{(0)}$ to neurons $x_{ik}^{(l)}$ as:

$$\boldsymbol{x}_i^{(l)} = \phi\left(\boldsymbol{h}_i^{(l)}\right), \quad \boldsymbol{h}_i^{(l)} = \sum_{j=1}^{c_{l-1}} \boldsymbol{W}_{ij}^{(l)} * \boldsymbol{x}_j^{(l-1)} + b_i^{(l)}, \quad (1)$$

where $\boldsymbol{h}^{(l)}$ is the pre-activation, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{c_l \times c_{l-1} \times k_l}$ is the weight tensor that consists of filters (or convolutional kernels), $\boldsymbol{b}^{(l)} \in \mathbb{R}^{c_l}$ is the bias vector of layer l, and $*$ denotes a convolution operation. To simplify and generalize our notation, we have flattened the filter dimension to k_l. For 2d convolutions, as they are commonly in use on imaging data, the weight tensor would actually have the size $\boldsymbol{W}^{(l)} \in$

$\mathbb{R}^{c_l \times c_{l-1} \times k'_{1,l} \times k'_{2,l}}$ so that $k_l = k'_{1,l} k'_{2,l}$. The convolution operation between any 2-dimensional tensors K and X is defined as $(\boldsymbol{K} * \boldsymbol{X})_{ij} = \sum_{i',j'} K_{i'j'} X_{(i-i'+1)(j-j'+1)}$ in this case. We assume that the inputs are always suitably padded with zeros and that the symbol $*$ performs the convolutions in the right dimensions. The flattened notation just makes it easier to discuss higher dimensional filters at the same time. In addition to convolutional layers, we also allow for residual and more general skip connections. Skip connections modify the network above as

$$\boldsymbol{x}_i^{(l)} = \phi\left(\boldsymbol{h}_i^{(l)}\right) + \sum_{t=1}^{l-1} \sum_{j=1}^{c_t} \boldsymbol{M}_{ij}^{(l,t)} * \boldsymbol{x}_j^{(t)}. \tag{2}$$

Usually, most of the operators $\boldsymbol{M}_{ij}^{(l,t)}$ are zero. In case of residual connections, $\boldsymbol{M}_{ij1}^{(l,t)} = 1$ are one-dimensional filters that encode the identity and do not impose any additional learnable or prunable parameters. Without loss of generality, we assume that each parameter (weight or bias) θ is bounded by $|\theta| \leq 1 - \epsilon$. In addition, we require that each tensor element is bounded as $|x_{iq}| \leq 1$. Otherwise, our estimate of the error that we allow in the approximation of each target parameter would become more complicated. Moreover, we denote with $N_{w,l}$ the number of all nonzero weight and bias parameters in Layer l that do not correspond to skip connections, while $N_{m,l}$ counts the number of all nonzero parameters involved in skip connections that lead to Layer l.

We distinguish three different types of neural networks: a target network f_t, a LT f_ϵ, and a source network f_0. The target network f_t is approximated by the LT f_ϵ, which we obtain by pruning f_0. We also write $f_\epsilon \subset f_0$, meaning that f_ϵ is constructed by masking some parameters of f_0, i.e. setting some of them to 0, while the other parameters keep their original value. The parameters of the source network f_0 are drawn from a random distribution as follows.

Assumption 2.1 (Parameter initialization)**.** We assume that the parameters of the source network f_0 are independently distributed as $w_{ij}^{(l)} \sim U\left([-\sigma_l, \sigma_l]\right)$, $b_i^{(1)} \sim U\left([-\sigma_l, \sigma_l]\right)$ and $b_i^{(l)} = 0$ for $l > 1$.

Note that they could also follow any other distribution that contains a uniform distribution, for instance, a normal distribution (Pensia et al., 2020). In our theorems and proofs, we choose σ_l conveniently based on the activation functions. For ReLUs, for instance, $\sigma_l = 1$ is common. In practice, we usually have $\sigma_l \propto c/\sqrt{k}$ to avoid vanishing or exploding gradients. To transfer our results to this setting, we need to scale each parameter of the LT of our proofs by a layer-wise scaling factor λ_l (Burkholz, 2022). For homogeneous activation functions like ReLUs or Leaky ReLUs, these scaling factors can also be joined into a single one $\lambda = \prod_l \lambda_l$ that is applied to the output (Fischer & Burkholz, 2021).

Note that if the target network parameter would not fulfill our assumption $|\theta| \leq 1$, we could simply adjust the scaling factors so that the initial parameter distribution and the network parameters vary within the same range. As we will see later, flexible scaling factors will also enable us to derive LT existence results for activation functions that can be approximated locally by a (leaky) ReLU without increasing our width requirement significantly. The remarkable reason is that we can control the approximation error by down-scaling the input parameters. Weak LT pruning algorithms learn these scaling factors automatically. Strong LT pruning algorithms would need to learn them in addition to the mask or adjust them based on our theoretical insights. Our existence results thus transfer to realistic parameter initialization settings.

As the parameters of the source network are random, f_0 needs to be bigger than the target so that we have enough alternatives to pick the right ones. Because of two strategies that help us increase our options, the LT also consists of more parameters and neurons than the target network. These two strategies are (a) solving subset sum approximation problems and (b) using several layers to approximate a target layer.

Subset Sum Approximation Instead of searching for a single parameter in f_0 that closely matches a target parameter θ_t, we approximate it by the sum of multiple parameters $\theta_t \approx \sum_{n \in S} \theta_{0,n}$. There are usually many options to represent such a sum and each possibility can be cast as a subset of a larger base set of size m, which contains 2^m candidate subsets. This explains why solving subset sum approximation problems to find a suitable subset is successful with high probability based on relatively small base sizes m. We frequently use a theorem by (Lueker, 1998), which has been extended by (Burkholz et al., 2022) to solve subset sum approximation problems if the random variables are not necessarily identically distributed. For convenience it is stated as Cor. A.2 in more general form in the appendix. We primarily utilize the following simplification in our construction.

Corollary 2.2 (Subset sum approximation (Lueker, 1998))**.** *Let $X_1, ..., X_m$ be independent, uniformly distributed random variables with $X_k \sim U[-1, 1]$ or $X_k \sim U[-1, 1]U[-1, 1]$ and $\epsilon, \delta \in (0, 1)$ be given. Then for any $\theta_t \in [-1, 1]$ there exists a subset $S \subset [m]$ so that with probability at least $1 - \delta$ we have $|\theta_t - \sum_{k \in S} X_k| \leq \epsilon$ if $m \geq C \log\left(\frac{1}{\min(\delta, \epsilon)}\right)$.*

Naturally, a question that decides about the practicality of this result is the size of the constant C. Repeated solutions of subset sum problems by optimal exhaustive searches over subsets for different base set sizes m with $X_k \sim U[-1, 1]$ suggest that $C \approx 3$, which means that our approach is feasi-

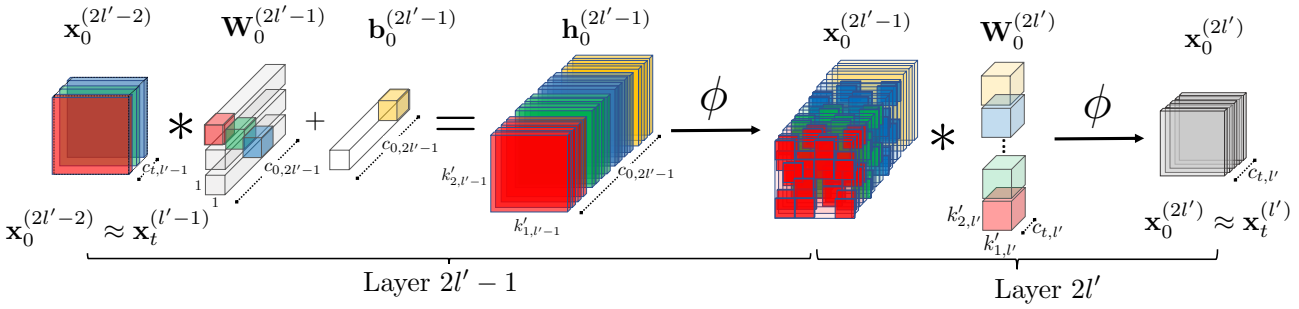

Figure 1. Construction of the target Layer l' in Layers $2l'-1$ and $2l'$ of the source network f_0.

ble. More insights on subset sum problems are discussed in the experiments section.

(b) The second strategy to construct LTs is concerned with making base sets $\{X_1, ..., X_m\}$ available to solve subset sum approximation problems. A common approach is to create multiple versions of the input to a layer, which usually populate an additional intermediary layer in f_0 and f_ϵ before approximate target neurons are constructed in the following layer. This results in source networks and LTs that need twice the depth of a target network, i.e, $L_0 = 2L_t$. (da Cunha et al., 2022) has transferred this idea for fully-connected feed forward architectures to convolutional layers with ReLU activation functions. However, the first layer has multi-dimensional input in general, which limits the approach to positive inputs. Our first contribution is to derive a different ($L_0 = 2L_t$)-construction that can handle any input and works for multiple activation functions, including ReLUs. Afterwards, we transfer the L_t+1 construction idea for fully-connected feed forward neural networks (Burkholz, 2022) and general activation functions to convolutional layers. An important consequence of this construction is that we can also cover residual blocks that are of the same size in all three networks, the target, the source, and the LT.

Activation Functions The ($L_0 = 2L_t$)-construction in fully-connected networks relies on the fact that ReLUs $\phi_R(x) = \max\{x, 0\}$ can easily represent the identity as $x = \phi_R(x) - \phi_R(-x)$ so that neurons in the intermediary layer correspond to the positive $\phi(x_i^{(l-1)})$ and the negative part $\phi(-x_i^{(l-1)})$ of input neurons $x_i^{(l-1)}$. Similarly, Leaky ReLUs $\phi_{LR}(x) = \phi_R(x) - \alpha\phi_R(-x)$ encode the identity as $x = (\phi_{LR}(x) - \phi_{LR}(-x))/(1+\alpha)$. Most other activation functions can be approximated locally around the origin by a shifted Leaky ReLU and thus fulfill the following.

Assumption 2.3 (Activation function (first layer))**.** For any given $\epsilon' > 0$ exists a neighborhood $[-a(\epsilon'), a(\epsilon')]$ of 0 with $a(\epsilon') > 0$ so that the activation function ϕ can be approximated by $\widehat{\phi}(x)$ on that neighborhood such that $\sup_{x\in[-a,a]} |\phi(x) - \widehat{\phi}(x)| \leq \epsilon'$, where $\widehat{\phi}(x) = m_+x + d$ for $x \geq 0$ and $\widehat{\phi}(x) = m_-x + d$ for $x < 0$ with $m_+, m_-, d \in \mathbb{R}$, and $m_+ + m_- \neq 0$. We further assume that, if $a(x)$ is finite, $g(x) = x/a(x)$ is invertible on an interval $]0, \epsilon'']$ with $\epsilon'' > 0$ and $\lim_{x\to 0} g(x) = 0$.

For instance, ReLUs $\phi(x) = \max(x, 0)$ inflict zero error on $\mathbb{R}$ (i.e., $a = \infty$) with $m_+ = 1$, $m_- = 0$, and $d = 0$. Leaky ReLUs can be represented without error with $m_+ = 1$, $m_- = \alpha$, and $d = 0$ for an $\alpha > 0$. $\phi(x) = \tanh(x)$ is approximately linear so that $|\tanh(x) - x| \leq x^3/3$ for $|x| < \pi/2$, which can be seen by Taylor expansion of tanh. This implies that the choice $m_+ = 1, m_- = 1$, and $d = 0$ with $a = \min\{(3\epsilon')^{1/3}, \pi/2\}$ fulfills our assumption. Sigmoids $\phi(x) = 1/(1 + \exp(-x))$ can be analyzed in the same way with $m_+ = m_- = 0.25$, $d = 0.5$, and $a = \min\{(48\epsilon')^{1/3}, \pi\}$, since $\phi(x) = (\tanh(x/2) + 1)/2$.

All these activation functions can approximate the identity as $|x - r\,(\phi(x) - \phi(-x))| \leq 2r\epsilon'$, where we have defined $r := \frac{1}{m_+ + m_-}$. To abbreviate our notation later on, we also use $\mu_\pm(x) := m_+$ for $x > 0$, $\mu_\pm(x) := m_-$ for $x < 0$, and $\mu_\pm(0) := 0$ for $x = 0$. Note that we always have $\mu_\pm(x) + \mu_\pm(-x) = m_+ + m_- = 1/r$. Functions with $m_+ = m_- = m$ and $d = 0$ like TANH can also be approximated by $|x - \phi(x)/m| \leq \epsilon'/m$ and do not need separate approximations of the positive and the negative part. The ability to easily approximate the identity is a valuable property to construct target networks that fit the depth of the source network. We can always increase the depth of a target by concatenating identity approximating layers. Note that we otherwise only need this assumption in the middle layers of the ($2L_t$)-construction and the first layer of the ($L_t + 1$)-construction. Any other activation functions do not even need to fulfill this assumption.

3. Existence Results

The main idea in our construction of LTs rests on the linearity of convolutions. Concretely, let us assume for a moment

that the input tensors in Eq. (1) of the LT are identical to a target input up to a scalar factor λ_j, i.e., $\boldsymbol{x}_{j'}^{(l-1)} = \lambda_{j'}\boldsymbol{x}_{t,j}^{(l-1)}$ for all $j' \in I_j$. It follows that $\sum_{j'\in I_j} \boldsymbol{W}_{ij'}^{(l)} * \boldsymbol{x}_{j'}^{(l-1)} = \sum_{j'\in I_j} \boldsymbol{W}_{ij'}^{(l)} * (\lambda_{j'}\boldsymbol{x}_{t,j}^{(l-1)}) = \left(\sum_{j'\in I_j} \boldsymbol{W}_{ij'}^{(l)}\lambda_{j'}\right) * \boldsymbol{x}_{t,j}^{(l-1)}$. We could therefore use $\left(\sum_{j'\in I_j} \boldsymbol{W}_{ij'}^{(l)}\lambda_{j'}\right)$ in the LT to approximate a target tensor $\boldsymbol{W}_{t,ij}^{(l)}$ by masking some of the components $w_{ij'q}^{(l)}$ according to subset sum approximation. In fact, all entries for the indices $j'q$ can be used independently to approximate the corresponding target tensor entry with index q. Hence, our main task is to create multiple candidates $\lambda_{j'}\boldsymbol{x}_{t,j}^{(l-1)}$ for an input $\boldsymbol{x}_{t,j}^{(l-1)}$. Our two different construction approaches differ in how they achieve this.

3.1. Two Layers for One

Informally, our first objective is to show that for any convolutional and/or residual target network f_t with depth L_t, maximum channel size c_t, and kernel size k_t, there exists with probability $1-\delta$ a sub-network f_ϵ of a source network with depth $L_0 = 2L_t$ that approximates the target up to error $\epsilon > 0$ if the source network has maximum channel size $c_0 \geq Cc_ts_0\log(c_tk_tL_t/\min\{\epsilon,\delta\}$ for a constant C that is independent of ϵ, δ, c_t, k_t, and the stride s_0 of filters in uneven layers of f_0.

Why do the source network and the LT have twice the depth? Fig. 1 visualizes the answer. Every neuron in an even layer $l = 2l'$ of the LT approximates the corresponding neuron in layer l' of the target so that $\boldsymbol{x}_{0,j}^{2l'} \approx \boldsymbol{x}_{t,j}^{l'}$ and $c_{0,2l'} = c_{t,l'}$ for every $l' \in [L_t]$. What happens in the uneven layers $l = 2l'+1$? We create multiple versions of the input neurons $\boldsymbol{x}_{0,j}^{(2(l'-1))}$ to support our subset sum approximation problems and we achieve this with the help of univariate filters. Note that the filters of f_0 do not need to be univariate themselves. We can also prune them into this state by setting all filter entries except for one to zero. Which filter entry we keep as nonzero depends on the stride. For simplicity, let us assume here that we have univariate filters $w_{i'j1}^{(2l'-1)}$ with stride $s_0 = 1$ already. How to transfer other cases to this setting is discussed in the appendix.

We have $w_{i'j1}^{(2l'-1)} = \lambda_{i'j}$ for every $i' \in I_j$ and $w_{i'j1}^{(2l'-1)} = 0$ otherwise. This creates preactivations $\boldsymbol{h}_{i'}^{(2l'-1)} = \sum_{j'} \boldsymbol{W}_{i'j'}^{(2l'-1)} * \boldsymbol{x}_{0,j'}^{(2l'-2)} = \lambda_{i'j}\boldsymbol{x}_{0,j}^{(2l'-2)}$, which is exactly what we were looking for. Yet, we still have to send each entry through an activation function receiving $\phi(\lambda_{i'j}\boldsymbol{x}_{0,j}^{(2l'-2)})$. To handle input entries with different signs, we have to create neurons with positive ($\lambda_{i'j} > 0$) and negative ($\lambda_{i'j} < 0$). For $|\lambda_{i'j}|$ small enough, we can then approximate ϕ as in Assumption 2.3, use this to construct the identity, and exchange the order of the summation and convolution because of the local linearity of ϕ. We receive $\sum_{i'\in I_j} \boldsymbol{W}_{0,ii'}^{(2l')} * \phi(\lambda_{i'j}\boldsymbol{x}_{0,j}^{(2l'-2)})$ $\approx \left(\sum_{i'\in I_j,\lambda_{i'j}>0} \boldsymbol{W}_{0,ii'}^{(2l')}\lambda_{i'j}\right)\left(\mu_\pm(\boldsymbol{x}_{0,j}^{(2l'-2)})\boldsymbol{x}_{0,j}^{(2l'-2)}\right) + \left(\sum_{i'\in I_j,\lambda_{i'j}<0} \boldsymbol{W}_{0,ii'}^{(2l')}\lambda_{i'j}\right) * \left(\mu_\pm(-\boldsymbol{x}_{0,j}^{(2l'-2)})\boldsymbol{x}_{0,j}^{(2l'-2)}\right)$ $\approx \boldsymbol{W}_{t,ij}^{(l')} * \left[r\left(\mu_\pm(-\boldsymbol{x}_{0,j}^{(2l'-2)}) + \mu_\pm(-\boldsymbol{x}_{0,j}^{(2l'-2)}\right)\boldsymbol{x}_{0,j}^{(2l'-2)}\right]$ $\approx \boldsymbol{W}_{t,ij}^{(l')} * \boldsymbol{x}_{t,j}^{(l'-1)}$, if we can approximate $r\sum_{i',\lambda_{i'j}>0} w_{0,ii'q}^{(2l')}\lambda_{i'j} \approx w_{t,ijq}^{(l')}$ and $r\sum_{i',\lambda_{i'j}<0} w_{0,ii'q}^{(2l')}\lambda_{i'j} \approx w_{t,ijq}^{(l')}$ for all filter entries q by appropriate masking of the tensor elements $w_{t,ijq}^{(l')}$. In addition, we have used that $r\left(\mu_\pm(-\boldsymbol{x}_{0,j}^{(2l'-2)}) + \mu_\pm(-\boldsymbol{x}_{0,j}^{(2l'-2)}\right) = 1$ by definition and that we have $\boldsymbol{x}_{0,j}^{(2l'-2)} \approx \boldsymbol{x}_{t,j}^{(l'-1)}$ by construction. Biases can be obtained similarly by approximating $\sum_{i',q} w_{0,ii'}^{(2l')}\mu_\pm(\boldsymbol{b}_{0,i'}^{(2l'-1)})\boldsymbol{b}_{0,i'}^{(2l'-1)} \approx \boldsymbol{b}_{t,i}^{(l')}$.

Theorem 3.1 (LT existence ($2L_t$-construction))**.** *Assume that $\epsilon, \delta \in (0,1)$, a convolutional target network (without skip connections) $f_t(x) : \mathcal{D} \subset \mathbb{R}^{c_0\times d_0} \to \mathbb{R}^{c_L\times d_{L_t}}$ with architecture $\bar{c}_t$ of depth L_t with $N_{t,l}$ nonzero parameters in Layer l, and a source network f_0 with architecture $\bar{n}_0$ of depth $L_0 = 2L_t$ are given. Let ϕ be the activation function of f_t with Lipschitz constant T fulfilling Assumption 2.3 with $d = 0$. Then, with probability at least $1-\delta$, f_0 contains a subnetwork $f_\epsilon \subset f_0$ so that each output component i is approximated as $\max_{\boldsymbol{x}\in\mathcal{D}} |f_{t,iq}(\boldsymbol{x}) - f_{\epsilon',iq}(\boldsymbol{x})| \leq \epsilon$, if for all $l' \in [L_t]$ we have*

$$c_{0,2l'+1} \geq Cc_{t,l}\log\left(\frac{N_t}{\min\{\epsilon/\prod_{s=l}^{L_t}(3TN_{t,s}),\delta\}}\right),$$

and $n_{0,2l'} \geq n_{t,l'} + 1$, and if the parameters of f_0 are initialized according to Assumption 2.1 with $\sigma_{2l'+1} = r/\sigma_{2l'}$ and $\sigma_{2l'} = a(\epsilon'')/2$ and $\epsilon'' = g^{-1}\left(\frac{\epsilon'}{CN_t\log\left(\frac{N_t}{\min\{\epsilon/\prod_{s=l}^{L_t}(3TN_{t,s}),\delta\}}\right)}\right)$ for $g(\epsilon'') = \epsilon''/a(\epsilon'')$.

The proof is provided in the appendix. In summary, three main insights enable the success of this construction and allow for generally positive and negative inputs in contrast to (da Cunha et al., 2022). First, we convolute the input channels with an univariate filter in the first layer (and not in the second as (da Cunha et al., 2022)). Second, the insight that we can then prune the entries of each filter in the second layer independently, which follows from the linearity of convolutions, makes the construction parameter efficient and flexible. Third, even if the activation function only approximates a (leaky) ReLU, we do not have to increase our width requirement significantly. The reason is that we

can scale the initial parameters in the first layer to become sufficiently small to allow for an accurate approximation of the activation function in a neighborhood of zero and compensate for this scaling in the second layer.

We could derive a more advantageous scaling of the error if we would additionally assume that $\left\|W_t^{(l)}\right\|_2 \leq 1$. Note also that the error does not depend on the input tensor dimension, i.e., the image size. This would change if we also incorporated a common flattening operation and fully-connected layers, which are often used to solve classification problems in the end. In this case, the image dimension would determine the number of input features to the fully-connected layers. As these are handled in a different work (Burkholz, 2022), we skip a deeper discussion. We would only need to adapt the initial ϵ to combine them.

We only state the theorem for activation functions $\phi(0) = d = 0$ here. For $d \neq 0$, we can derive similar results if we change the initialization scheme to 'looks-linear' initialization (Burkholz & Dubatovka, 2019) to control the error when we approximate ϕ, see also (Burkholz, 2022).

3.2. ($L_0 = L_t + 1$)-Construction

The $(2L_t)$-construction uses an additional layer to create multiple versions of the input for each target layer. As the neuron states are sent through the non-linear activation function, we usually need to create two neurons, the analog to the positive and the analog to the negative part of each input channel. Yet, this extra effort would not be necessary if we had immediately multiple versions of each input channel available. As we cannot change the given input by the data, to create multiple versions initially, we have to employ the two-for-one layer construction to approximate the first target layer. If we directly construct the $c_{t,1}$ output channels multiple times, we can however drop the next intermediary layer completely and repeat constructing the channels of the next layer so many times that they serve the solution of subset sum approximation problems in the following layers. Fig. 2 explains the main idea and the next theorem states formally the existence result for this construction.

Theorem 3.2 (LT existence ($2L_t$-construction))**.** *Assume that $\epsilon, \delta \in (0,1)$, a convolutional target network (possibly with skip connections) $f_t(x) : \mathcal{D} \subset \mathbb{R}^{c_0 \times d_0} \to \mathbb{R}^{c_L \times d_{L_t}}$ with architecture $\bar{c}_t$ of depth L_t, and a source network f_0 with architecture $\bar{n}_0$ of depth $L_0 = L_t + 1$ are given. Let ϕ be the activation function of f_t and f_0 with Lipschitz constant T. Furthermore, let ϕ_0 be the activation function of f_0 in the first layer fulfilling Assumption 2.3 with $d = 0$. Define the number N_l of effective nonzero parameters in Layer l as $N_l = N_{w,l} + N_{m,l}$. Then, with probability at least $1 - \delta$, f_0 contains a subnetwork $f_\epsilon \subset f_0$ so that each output component is approximated as $\max_{\boldsymbol{x} \in \mathcal{D}} |f_{t,iq}(\boldsymbol{x}) - f_{\epsilon',iq}(\boldsymbol{x})| \leq \epsilon$*

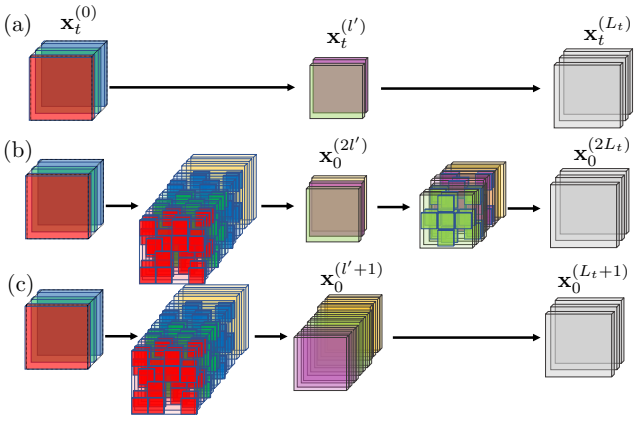

Figure 2. Comparison of LT construction approaches. (a) Target network f_t. (b) $(2L_t)$-construction. (c) $(L_t + 1)$ construction.

if for all $l \in [L_t]$ we have

$$c_{0,l+1} \geq C c_{t,l} \log\left(\frac{1}{\min\{\epsilon_l, \rho\delta/N_l\}}\right),$$

and $n_{0,1} \geq C c_{t,0} \log\left(\frac{1}{\min\{\epsilon_1, \delta\rho\}}\right)$ *with* $\rho = C N_l^{1+\gamma} \log(1/\min\{\min_l \epsilon_l, \delta\})$ *for any* $\gamma > 0$. $\epsilon_l = \frac{\epsilon}{2TN_{w,l}\prod_{s=l+1}^{L}(2(TN_{w,s}+N_{m,s}))}$. *Additionally, we require that the parameters of f_0 are initialized according to Assumption 2.1 with $\sigma_l = 1$ for $l > 2$, $\sigma_1 = r/\sigma_2$ and $\sigma_2 = a(\epsilon'')/2$ and suitably chosen ϵ''.*

The proof is given in the appendix. The main challenge in the derivation is to identify the size of the required subset sum blocks, as it depends on the number of total subset sum problems that need to be solved. This number in turn depends on the subset sum block sizes. Both need to be balanced as stated in the theorem. We observe that the block size is potentially larger in the $(L_t + 1)$-construction than in the $(2L_t)$-construction if more subset sum problems need to be solved to create multiple versions of the previous channel directly. But this factor enters only the logarithm and is thus small. The fact that the $(2L_t)$-construction requires often double the amount of channels to regard the analogs of the positive and negative part separately, is usually a stronger requirement. For a very high number of target parameters, the required L_t+1 blocks might still be larger. Furthermore, the LT in the $L_t + 1$-construction consists of many more parameters, yet, less neurons, which is often dominating the computational needs on GPUs. We discuss these differences in detail in the experiments section. Regardless of the advantages and disadvantages of each construction, the main purpose of this theorem is to show that LTs exist that can leverage most of the source network's depth. Which construction could be found by different pruning algorithms and which one would be better is a different question.

4. Experiments

Our theoretical insights suggest that source networks do not need to be much larger than the networks that we want to approximate by a lottery ticket - at least with regard to how our width requirement scales with the relevant parameters. Three possible challenges could still arise in practice. First, the size of the constant in our width requirement might be impractically large. Second, deep target networks consisting of many parameters might be very fragile to small errors in their parameters so that ϵ_l is so small that even $\log(1/\epsilon_l)$ in our width requirement is practically too large. Similarly, it could be the case that we would need to solve so many subset sum problems, in particular in the one-layer-for-one construction, that δ/N would be too small. The following experiments rule out these three concerns and show that constructing lottery tickets by solving subset sum approximations is practically feasible.

In fact, our proofs define implicitly an algorithm that approximates a target network by pruning a source network consisting of $m(c_{t,l}+1)+c_0$ channels in each convolutional layer with $l>1$, $2m_2(c_{t,0}+1)+c_0$ channels in the first layer, and $c_{t,L}$ channels in the output layer. m here refers to the size of subset sum blocks that we choose. c_0 gives us the option to fail sometimes in solving a subset sum problem and continue with pruning another neuron instead. It is negligible in our experiments.

Thus, our first question has to be: How large should we choose m (and m_2)? Even though solving subset sum problems is in general NP-hard, for small enough m we can still solve them optimally by exhaustively evaluating all subsets. Fig. 3 (a) presents statistics on these optimal solutions for $m=15$, which is generally considered to be sufficiently large in the literature. On average, the error within the 95% standard confidence interval is $(6\pm0.8)10^{-4}$, but in most cases it is much smaller, as the left skewed distribution indicates. It exceeds 0.01 only in 0.3% of the cases.

In addition to the approximation error, we are also interested in the number of parameters that enter our LT and thus the size of the selected subset $|S|$. If our target layer consists of N_l nonzero parameters, we expect that our LT consists on average of $m\mathbb{E}(|S|)N_l$ parameters. Fig. 3 (b) shows the distribution of $|S|$, which has $\mathbb{E}(|S|)\approx 7$. For increasing m, the average error decreases but the average subset size would increase, which is not ideal for constructing sparse tickets. However, LTs do not require the optimal subset. A subset that reaches a small enough error would be sufficient. Fig. 3 (c-d) show therefore statistics for a subset selection process that searches through all subsets but stops when it finds a subset whose approximation error does not exceed 0.01. In this case, we only need $\mathbb{E}(|S|)\approx 2.3$. This observation is also relevant for the LT's sparsity assessment. Commonly, sparsity is reported relative to a dense network. Thus, if the target layer has sparsity $\rho_t = N_l/(c_{t,l}c_{t,l-1}K)$, the LT has roughly $\rho_\epsilon = m\mathbb{E}(|S|)N_l/([m(c_{t,l}+1)+c_0][m(c_{t,l-1}+1)+c_0]K)\approx \rho_t\mathbb{E}(|S|)/m$. Accordingly, the choice $m=15$ and allowed error $\epsilon_l=0.01$ let's us construct LTs that have 15% of the target's sparsity. We could always reduce this number further by equipping the source network with a higher number of channels that we can prune away. The real quantity of interest is therefore the total number of required nonzero parameters, which we report in the following experiments alongside the neural network performance.

Table 1. Test accuracy in % of pruned LTs and their $L=3$ target on MNIST for $(L+1)$ construction. Averages and 0.95 standard confidence intervals are reported for 10 repetitions.

	TARGET	LT
RELU	98.8	98.72 ± 0.04
LEAKY RELU	98.5	98.5 ± 0.03
TANH	98.14	98.09 ± 0.09
SIGMOID	98.52	98.5 ± 0.004

We could repeat a similar analysis for the two-layers-for-one construction. In this case, we would have to solve subset sum problems with random variables that are distributed as the product of two uniform random variables $X_k = U[-1,1]U[-1,1]$. Statistics regarding this case are presented in the appendix in Fig. 4. The overall distributions look similar. A base set size of $m=15$ random variables is also sufficient but results in an average error of $(7.6\pm0.3)10^{-3}$, which is more than a magnitude higher than in the previous case. Furthermore, we fail to achieve an error smaller than 0.01 in 3.5% of the cases and rely on subset sizes of $\mathbb{E}(|S|)\approx 2.5$. This only affects the first layer of our experiments though.

To demonstrate that an error of $\epsilon_l=0.01$ per parameter is indeed acceptable to obtain LTs most of the time, we employ the described pruning strategy that solves subset sum approximation problems with small subsets that try to achieve an error of maximally $\epsilon_l=0.01$ with respect to each parameter of a given target network. We identify two different types of target networks with the established Synflow algorithm and its open source code (Tanaka et al., 2020), which we apply with exponential annealing of the target sparsity with 12 steps on a machine with Intel(R) Core(TM) i9-10850K CPU @ 3.60GHz processor and GPU NVIDIA GeForce RTX 3080 Ti. Between pruning steps, we train the pruned network for 50 epochs.

The first target network type is a small scale example with 3 layers and is pruned and trained on MNIST (Deng, 2012). Its first two layers are convolutional with 32 channels and 3x3 filters before pruning. The last layer is a fully-connected

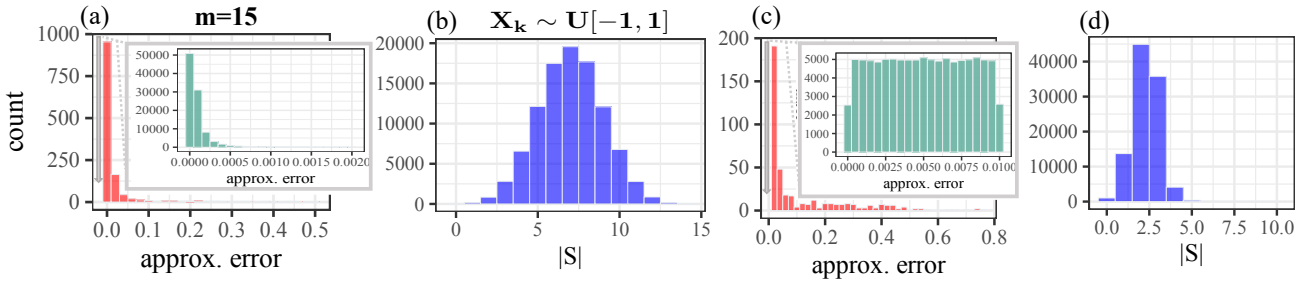

Figure 3. Subset sum approximation statistics. (a) & (c): Error of solving 10^5 independent subset sum problems. Each problem selects $|S|$ elements out of $m = 15$ independent random variables $X_k \sim U[-1, 1]$ to approximate a randomly drawn target $z \sim U[-1, 1]$. The green histogram in the right corner focuses on the smallest errors. (b) & (d): Size of approximating subset. (a) & (b): Best subset selection based on exhaustive search. (c) & (d): Smallest subset that achieves an error of maximally $\epsilon = 0.01$.

Table 2. Number of prunable neural network parameters for experiments reported in Table 1 regarding the MNIST example.

	TARGET	LT
RELU	945	4386 ± 34
LEAKY RELU	940	4545 ± 44
TANH	950	3269 ± 19
SIGMOID	953	4198 ± 59

Table 3. Test accuracy in % of pruned LTs and their target for ResNet-22 on CIFAR10 for $(L + 1)$ construction. Averages and 0.95 standard confidence intervals are reported for 50 repetitions.

	TARGET	LT	LT (BEST 50%)
RELU	80.68	80.32 ± 0.13	80.69 ± 0.09
LEAKY RELU	80.68	80.11 ± 0.18	80.6 ± 0.1
TANH	80.86	80.48 ± 0.13	80.8 ± 0.1
SIGMOID	72.66	69 ± 2	73.2 ± 0.9

Table 4. Number of prunable neural network parameters for experiments reported in Table 3 regarding the ResNet-22 example.

	TARGET	LT
RELU	31069	1058140 ± 559
LEAKY RELU	31113	1059537 ± 567
TANH	64491	2193086 ± 723
SIGMOID	116048	3934621 ± 1248

classification layer with softmax activation functions. We obtain separate targets for four commonly used activation functions in the first two layers: RELU, LEAKY RELU, SIGMOID, and TANH and explicitly approximate the convolutional layers with an $L + 1$ construction by solving the associated subset sum problems. The performance of the resulting neural networks is reported in Table 1 and the number of the pruned nonzero parameters in Table 2. We observe a very similar performance of the LTs in comparison with the target networks. While we report averages over 50 independent runs, note that we can always find a couple of lucky solutions that outperform the target network on the test set. If we would use the optimal subset instead of a small one, we would see no significant difference between the target and the LT. Yet, the resulting LT would also consist of more parameters. The delicate trade-off between LT size and potential accuracy has to be solved in practice.

The second target that we consider has a more realistic structure that is much deeper and includes residual blocks. Similar to before, we prune and train ResNet-22 on CIFAR-10 (Krizhevsky, 2009). To save computational time, we draw for each parameter and error and associated set size from the empirical distribution that we derived by solving 10^5 independent subset sum problems as shown in Figure 3 (c-d) (and the appendix for the first layer). The results are presented in Tables 3&4.

5. Conclusions

We have proven that pruning randomly initialized convolutional neural networks, including residual blocks and skip connections, can be a viable strategy to identify smaller scale networks. These lottery tickets (LTs) can be found if they approximate target networks whose width is smaller by a logarithmic factor than the original random source network. In practice, a factor of $1/15$ is sufficient, as we have verified in experiments. Our proofs are the first to cover residual and skip connections and other activation functions than ReLUs for convolutional layers. We have furthermore presented a novel LT construction for convolutional layers that is not restricted to positive inputs and discussed two versions. The first one assumes that the depth of the target network is $L_t \leq L_0/2$. This construction is more parameter efficient but requires a relatively high depth and usually a

higher number of neurons than the second version. In the second version, the target depth can only be slightly smaller than the depth of the large random network $L_t \leq L_0 - 1$. While the resulting LT consists of many more parameters relative to the target representation, the target representation itself can be much sparser, as it is allowed to utilize most of the available depth L_0. Furthermore, this construction indicates that not only extremely deep networks contain LTs. This is an important finding, as we can thus focus our pruning efforts on neural networks of similar depth as contemporary architectures, which are feasible to train.

Reproducibility and Code Availability

Pseudocode for the algorithm that constructs a LT to approximate a given target network is provided in Appendix F. The Github repository RelationalML/LT-existence contains a corresponding Pytorch implementation and all code for the experiments.

References

Burkholz, R. Most activation functions can win the lottery without excessive depth. In *arXiv*, 2022.

Burkholz, R. and Dubatovka, A. Initialization of ReLUs for dynamical isometry. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Burkholz, R., Laha, N., Mukherjee, R., and Gotovos, A. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022.

Chen, T., Cheng, Y., Gan, Z., Liu, J., and Wang, Z. Data-efficient GAN training beyond (just) augmentations: A lottery ticket perspective. In *Advances in Neural Information Processing Systems*, 2021a.

Chen, T., Sui, Y., Chen, X., Zhang, A., and Wang, Z. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, 2021b.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Liu, J., and Wang, Z. The elastic lottery ticket hypothesis. In *Advances in Neural Information Processing Systems*, 2021c.

da Cunha, A., Natale, E., and Viennot, L. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2022.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Diffenderfer, J. and Kailkhura, B. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. In *International Conference on Learning Representations*, 2021.

Dong, X., Chen, S., and Pan, S. J. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 2017.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2020.

Fischer, J. and Burkholz, R. Towards strong pruning for lottery tickets with non-zero biases, 2021.

Fischer, J. and Burkholz, R. Plant 'n' seek: Can you find the winning ticket? In *International Conference on Learning Representations*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015.

Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *International Conference on Neural Information Processing Systems*, 1992.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems*, 1990a.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990b.

Lee, N., Ajanthan, T., and Torr, P. H. S. Snip: single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

Lee, N., Ajanthan, T., Gould, S., and Torr, P. H. S. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Liu, N., Yuan, G., Che, Z., Shen, X., Ma, X., Jin, Q., Ren, J., Tang, J., Liu, S., and Wang, Y. Lottery ticket preserves weight correlation: Is it desirable or not? In *International Conference on Machine Learning*.

Liu, N., Yuan, G., Che, Z., Shen, X., Ma, X., Jin, Q., Ren, J., Tang, J., Liu, S., and Wang, Y. Lottery ticket preserves weight correlation: Is it desirable or not? In *International Conference on Machine Learning*, 2021a.

Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., and Mocanu, D. C. Sparse training via boosting pruning plasticity with neuroregeneration. In *Advances in Neural Information Processing Systems*, 2021b.

Lueker, G. S. Exponentially small bounds on the expected optimum of the partition and subset sum problems. *Random Structures & Algorithms*, 12(1):51–62, 1998.

Ma, X., Yuan, G., Shen, X., Chen, T., Chen, X., Chen, X., Liu, N., Qin, M., Liu, S., Wang, Z., and Wang, Y. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? In *Advances in Neural Information Processing Systems*, 2021.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, 2020.

Mhaskar, H., Liao, Q., and Poggio, T. When and why are deep networks better than shallow ones? In *AAAI Conference on Artificial Intelligence*, pp. 2343–2349, 2017.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

Mozer, M. C. and Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, 1989.

Orseau, L., Hutter, M., and Rivasplata, O. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33, 2020.

Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2599–2610, 2020.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems*, 2020a.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems*, 2020b.

Srinivas, S. and Babu, R. V. Generalized dropout. *CoRR*, abs/1611.06791, 2016.

Su, J., Chen, Y., Cai, T., Wu, T., Gao, R., Wang, L., and Lee, J. D. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Advances in Neural Information Processing Systems*, 2020.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*, 2020.

Verdenius, S., Stol, M., and Forré, P. Pruning via iterative ranking of sensitivity statistics, 2020.

Wang, C., Zhang, G., and Grosse, R. B. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

Weigend, A., Rumelhart, D., and Huberman, B. Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems*, 1991.

Yarotsky, D. Optimal approximation of continuous functions by very deep relu networks. In *Conference On Learning Theory*, pp. 639–649, 2018.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*, 2020.

Zhang, S., Wang, M., Liu, S., Chen, P.-Y., and Xiong, J. Why lottery ticket wins? a theoretical perspective of sample complexity on sparse neural networks. In *Advances in Neural Information Processing Systems*, 2021a.

Zhang, Z., Chen, X., Chen, T., and Wang, Z. Efficient lottery ticket finding: Less data is more. In *International Conference on Machine Learning*, 2021b.

Zhang, Z., Jin, J., Zhang, Z., Zhou, Y., Zhao, X., Ren, J., Liu, J., Wu, L., Jin, R., and Dou, D. Validating the lottery ticket hypothesis with inertial manifold theory. In *Advances in Neural Information Processing Systems*, 2021c.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pp. 3597–3607, 2019.

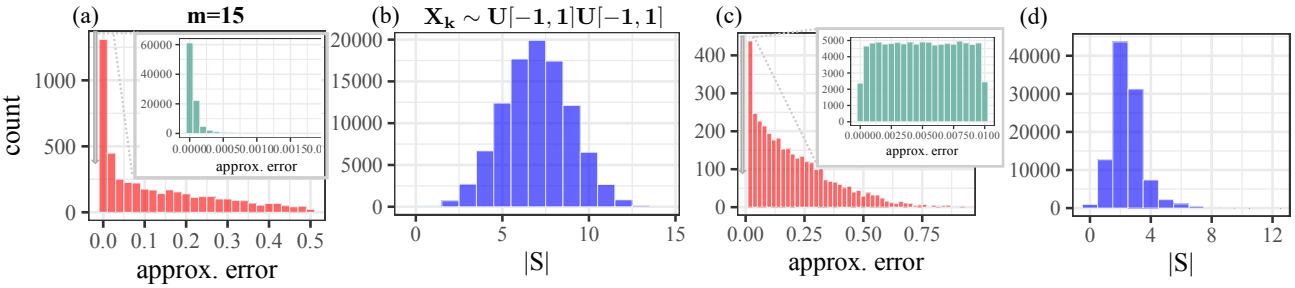

Figure 4. Subset sum approximation statistics. (a) & (c): Error of solving 10^5 independent subset sum problems. Each problem selects $|S|$ elements out of $m = 15$ independent random variables $X_k \sim U[-1, 1]U[-1, 1]$ to approximate a randomly drawn target $z \sim U[-1, 1]$. The green histogram in the right corner focuses on the smallest errors. (b) & (d): Size of approximating subset. (a) & (b): Best subset selection based on exhaustive search. (c) & (d): Smallest subset that achieves an error of maximally $\epsilon = 0.01$.

A. Subset Sum Approximation

We generally have multiple random neurons and parameters available to approximate a target parameter z by $\widehat{z}$ up to error ϵ so that $|z - \widehat{z}| \leq \epsilon$. Let us denote these random parameters in the source network as X_i. If these contain a uniform distribution, as defined below, we can utilize a subset of them for approximating z.

Definition A.1. A random variable X contains a uniform distribution if there exist constants $\alpha \in (0, 1]$, $c, h > 0$ and a random variable G_1 so that X is distributed as $X \sim \alpha U[c - h, c + h] + (1 - \alpha)G_1$.

(Burkholz et al., 2022) extended results by (Lueker, 1998) to solve subset sum approximation problems if the random variables are not necessarily identically distributed. In addition, they also cover the case $|z| > 1$. The general statement follows below.

Corollary A.2 (Subset sum approximation (Lueker, 1998; Burkholz et al., 2022))**.** *Let $X_1, ..., X_m$ be independent bounded random variables with $|X_k| \leq B$. Assume that each $X_k \sim X$ contains a uniform distribution with potentially different $\alpha_k > 0$ (see Definition A.1) and $c = 0$. Let $\epsilon, \delta \in (0, 1)$ and $t \in \mathbb{N}$ with $t \geq 1$ be given. Then for any $z \in [-t, t]$ there exists a subset $S \subset [m]$ so that with probability at least $1 - \delta$ we have $|z - \sum_{k \in S} X_k| \leq \epsilon$ if*

$$m \geq C \frac{\max\left\{1, \frac{t}{h}\right\}}{\min_k\{\alpha_k\}} \log\left(\frac{B}{\min\left(\frac{\delta}{\max\{1, t/h\}}, \frac{\epsilon}{\max\{t, h\}}\right)}\right).$$

B. Additional Statistics on Solving Subset Sum Problems

In the 2-layers-for-one construction, we solve subset sum approximation problems based on independent random variables with distribution $X_k \sim U[-1, 1]U[-1, 1]$, i.e., the product of two random variables. To be more precise, for ReLUs, the random variables are distributed as $X_k \sim U[0, 1]U[-1, 1]$ or $X_k \sim U[-1, 0]U[-1, 1]$. Because of the symmetry of the uniform random variables, these are all identically distributed and we can just focus on the case $X_k \sim U[-1, 1]U[-1, 1]$. Fig. 4 shows the corresponding statistics that are based on 10^5 independent subset sum problem solutions for base sets of size $m = 15$.

A base set size of $m = 15$ random variables results in an average error of $(7.6 \pm 0.3)10^{-3}$, which is more than a magnitude higher than in the case of $X_k \sim U[0, 1]$. Furthermore, we fail to achieve an error smaller than 0.01 in 3.5% of the cases and rely on subset sizes of $\mathbb{E}(|S|) \approx 2.5$.

C. Proofs

We frequently use the supremum norm of tensors, which is similarly defined as a vector norm refers to the maximum absolute value of all components $\|X\|_\infty := \max_{i,j,k} |x_{ijk}|$. It should not be confused with the operator norm.

C.1. Proof of Thm. 3.1

The following corollary covers an important step in the proof of our $(2L)$-construction, as it focuses on the approximation of a single layer by two layers in the source network.

Corollary C.1 (Layerwise approximation)**.** *Let a convolutional target layer $\boldsymbol{x}_{t,i}^{(l')} = \phi\left(\sum_{j=1}^{c_{l-1}} \boldsymbol{W}_{t,ij}^{(l')} * \boldsymbol{x}_{t,j}^{(l'-1)} + b_{t,i}^{(l')}\right)$ with $c_{l'}$ output, $c_{l'-1}$ input channels, and filter size $k_{l'}$, N_t nonzero parameters and activation function fulfilling Assumption 2.3 with $d = 0$ be given. Moreover, we have a two-layer source network $\boldsymbol{x}_{0,i}^{(2l')} = \phi\left(\sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{0,is}^{(2l')} * \phi\left(\sum_{j=1}^{c_{l'-1}} \boldsymbol{W}_{0,sj}^{(2l'-1)} * \boldsymbol{x}_{0,j}^{(2l'-2)}\right)\right)$ with parameters $\boldsymbol{W}_0^{(2l')} \in \mathbb{R}^{c_{l'} \times c_{1/2} \times k_{l'}}$ and $\boldsymbol{W}_0^{(2l'-1)} \in \mathbb{R}^{c_{1/2} \times c_{l'-1} \times 1}$. All its tensor entries are independently uniformly distributed as $w_{0,sjq}^{(2l'-1)} \sim U[-\sigma, \sigma]$ and $w_{0,isq}^{(2l')} \sim U[-r/\sigma, r/\sigma]$. Then, for every $\epsilon', \delta' \in (0,1)$, with probability $1-\delta'$, there exists a sub-network $\boldsymbol{x}_{\epsilon'}^{(2l')} \subset \boldsymbol{x}_0^{(2l')}$ so that $\left\|\boldsymbol{x}_t^{(l')} - \boldsymbol{x}_{\epsilon}^{(2l')}\right\|_\infty \leq \epsilon'$ if*

$$c_{1/2} \geq C c_{l'-1} \log\left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}}\right), \tag{3}$$

the input components fulfill $\left\|\boldsymbol{x}_{t,i}^{(l'-1)} - \boldsymbol{x}_{0,i}^{(2l'-2)}\right\|_\infty \leq \epsilon'/(3TN_t)$, $\left\|1 - \boldsymbol{x}_{0,c_{l'-1}+1}^{(2l'-2)}\right\|_\infty \leq \epsilon'/(3TN_t)$, and $\sigma = \min\{1, a(\epsilon'')/2\}$ with

$$\epsilon'' = g^{-1}\left(\frac{\epsilon'}{3TN_t C \log\left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}}\right)\frac{r}{2}}\right)$$

for $g(\epsilon'') = \epsilon''/(a(\epsilon''))$.

Proof. The first step of our construction of the LT $\boldsymbol{x}_{\epsilon'}^{(2l')}$ is to prune the weight tensors in the first layer of the source network $\boldsymbol{W}_{0,js}^{(2l'-1)}$ to univariate form. For each input neuron j, we reserve $|I_j|$ neurons in the intermediary layer $2l'-1$ with indices I_j so that $w_{\epsilon',sj1}^{(2l'-1)} = w_{0,sj1}^{(2l'-1)} = \lambda_{sj}$ if $s \in I_j$ and $w_{\epsilon',sj1}^{(2l'-1)} = 0$ otherwise. After pruning, we thus have $\phi\left(\sum_{j=1}^{c_{l'-1}} \boldsymbol{W}_{\epsilon',sj}^{(2l'-1)} * \boldsymbol{x}_{0,j}^{(2l'-2)}\right) = \phi\left(\boldsymbol{W}_{\epsilon',sj}^{(2l'-1)} * \boldsymbol{x}_{0,j}^{(2l'-2)}\right) = \phi\left(\lambda_{sj}\boldsymbol{x}_{0,j}^{(2l'-2)}\right)$ for an $s \in I_j$.

Per construction of the initialization, $\sigma > 0$ is chosen small enough so that

$$\left\|\phi\left(\lambda_{sj}\boldsymbol{x}_{0,j}^{(2l'-2)}\right) - \mu_\pm(\lambda_{sj}\boldsymbol{x}_{0,j}^{(2l'-2)})\lambda_{sj}\boldsymbol{x}_{0,j}^{(2l'-2)}\right\|_\infty \leq \epsilon'' \tag{4}$$

according to Assumption 2.3. We achieve this, as $|\lambda_{sj}x_{0,jq}^{(2l'-2)}| \leq \sigma(1+\epsilon'/(3TN_t)) \leq a(\epsilon'')$ with $|x_{0,jq}^{(2l'-2)}| \leq 1 + \epsilon'/(3TN_t)$, since $|x_{t,jq}^{(l'-1)}| \leq 1$ is always assumed.

The next step of pruning is to mask some elements of the tensor in the second layer $\boldsymbol{W}_{0,is}^{(2l')}$ so that we can approximate our target parameters. We set all parameters to zero $w_{\epsilon',isq}^{(2l')} = 0$ with the exception of $w_{\epsilon',isq}^{(2l')} = w_{0,isq}^{(2l')}$ for indices $s \in I_{ijq} \subset I_j$. These index sets are chosen by solving specific subset sum approximation problems based on the following two base sets $I_{j,+}$ and $I_{j,-}$ with $I_j = I_{j,+} \cup I_{j,-}$. They are defined according to the incoming link weight λ_{sj} in the first layer, i.e. $I_{j,+} = \{s \in I_j \mid \lambda_{sj} > 0\}$ and $I_{j,-} = \{s \in I_j \mid \lambda_{sj} < 0\}$. The associated random variables $X_s = w_{0,isq}^{(2l')}\lambda_{ij}/r$ are distributed as $U[0,1]U[-1,1]$ or $U[-1,0]U[-1,1]$ per construction. As these contain uniform distributions (Pensia et al., 2020), according to Thm. A.2, with probability $1-\delta'''$ we can find subsets $I_{ijq}^\pm \subset I_{j,\pm}$ so that

$$|w_{t,ijq} - \sum_{s \in I_{ijq}^\pm} X_s| \leq \epsilon''', \tag{5}$$

if $|I_{j,\pm}| \geq C \log\left(\frac{1}{\min\{\epsilon''', \delta'''\}}\right)$. Solving two separate problems of this form leads to an index set $I_{ijq} = I_{ijq}^+ \cup I_{ijq}^-$ that defines the parameters that we keep in our LT.

We still have to approximate the target bias. For that purpose, we have reserved a constant input tensor $\boldsymbol{x}_{0,c_{l'-1}+1}^{(2l'-2)}$ with $x_{0,c_{l'-1}+1,q}^{(2l'-2)} \approx 1$ so that $\phi(\lambda_{s(c_{l'-1}+1)} x_{0,c_{l'-1}+1,q}^{(2l'-2)}) \approx m_+ \lambda_{s(c_{l'-1}+1)}$. We thus choose nonzero parameters with indices $I_{ib} \subset I_{c_{l'-1}+1}$ in the second layer that solve the subset sum approximation problem

$$|b_{t,i} - \sum_{s \in I_{ib}} X_s| \leq \epsilon''', \tag{6}$$

as the random variables $X_s = w_{0,is}^{(2l')} \lambda_{s(c_{l'-1}+1)} m_+$ are distributed as $m_+ r U[0,1] U[-1,1]$ (with $|m_+ r| \leq 1$ most of the time), which also contain a uniform distribution.

After pruning the first and the second layer of the source network this way, let us analyze the error that our LT inflicts. It follows from the Lipschitz continuity of the activation function ϕ and our pruning to univariate tensors in the first layer that

$$\begin{aligned}
&\left\| \boldsymbol{x}_t^{(l')} - \boldsymbol{x}_\epsilon^{(2l')} \right\|_\infty \leq T \max_i \sum_j \left\| \boldsymbol{W}_{t,ij}^{(l')} * \boldsymbol{x}_{t,j}^{(l'-1)} + b_{t,i}^{(l')} - \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \phi\left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)}\right) \right\|_\infty \\
&\leq T \max_i \sum_j \left\| \boldsymbol{W}_{t,ij}^{(l')} * \boldsymbol{x}_{t,j}^{(l'-1)} + b_{t,i}^{(l')} - \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left(\mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{t,j}^{(l'-1)}\right) \lambda_{sj} \boldsymbol{x}_{t,j}^{(l'-1)}\right) \right\|_\infty \\
&+ T \max_i \sum_j \left\| \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left(\mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)}\right) \lambda_{sj} \left[\boldsymbol{x}_{0,j}^{(2l'-2)} - \boldsymbol{x}_{t,j}^{(l'-1)}\right]\right) \right\|_\infty \\
&+ T \max_i \sum_j \left\| \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left[\mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)}\right) \lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)} - \phi\left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)}\right)\right] \right\|_\infty \leq \epsilon'
\end{aligned} \tag{7}$$

Note that $\mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)}\right) = \mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{t,j}^{(l'-1)}\right)$ if $|\boldsymbol{x}_{t,j}^{(2l'-2)}| > \epsilon'/(3TN_t)$ anyways. Otherwise, we would prune it to zero. The first term concerns the subset sum approximation error, the second one the approximation of the input neurons, while the third one originates in the approximation of the activation function in the first layer. We achieve our approximation objective if we bound each of these errors by $\epsilon'/3$ Note that the latter vanishes for ReLUs, Leaky ReLUs, or linear activation functions because our approximation would actually be exact.

Let us first bound the subset sum approximation error. Recall that we have assumed that $|x_{t,jq}^{(l'-1)}| \leq 1$. We can partition the sum over the indices s to focus on the same input so that we can utilize the linearity of convolutions to derive

$$\begin{aligned}
&T \max_i \sum_j \left\| \boldsymbol{W}_{t,ij}^{(l')} * \boldsymbol{x}_{t,j}^{(l'-1)} + b_{t,i}^{(l')} - \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left(\mu_\pm\left(\lambda_{sj} \boldsymbol{x}_{t,j}^{(l'-1)}\right) \lambda_{sj}\right) \right\|_\infty \\
&= T \max_i \sum_j \sum_q \left(\left| r \mu_\pm\left(x_{t,jq}^{(l'-1)}\right) \left(\boldsymbol{W}_{t,ijq}^{(l')} - 1/r \sum_{s \in I_{ijq}^+} w_{\epsilon',isq}^{(2l')} \lambda_{sj} \right) \right| \right. \\
&\left. + \left| r \mu_\pm\left(-x_{t,jq}^{(l'-1)}\right) \left(\boldsymbol{W}_{t,ijq}^{(l')} - 1/r \sum_{s \in I_{ijq}^-} w_{\epsilon',isq}^{(2l')} \lambda_{sj} \right) \right| + \left| b_{t,i}^{(l')} - m_+ \sum_{s \in I_{ib}} w_{0,is}^{(2l')} \lambda_{s(c_{l'-1}+1)} \right| \right) \leq T N_t \epsilon''' \leq \epsilon'/3,
\end{aligned} \tag{8}$$

where we have used Eqs. (5) & (6) and the fact that $r(\mu_\pm(x) + \mu_\pm(-x)) = 1$. Defining $\epsilon''' = \epsilon'/(3TN_t)$ derives our width requirement so that we can solve the associated subset sum approximation problems. How should we choose δ'''? In total, we have to solve less than $2N_t$ problems. If each is successfully solved with probability $1 - \delta''' = 1 - \delta'/(2N_t)$, we can see with the help of a union bounds that we can solve all of them with probability $(1 - \delta')$. For every input neuron, we need to prune a large enough base set with $||I_j| \geq C \log\left(\frac{1}{\min\{\epsilon''', \delta'''\}}\right)$ resulting in a width requirement of $c_{1/2} \geq C c_{l'-1} \log\left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}}\right)$, as was to be shown.

Second, we have to show that we can bound the approximation error of the input neurons in Eq. (7). With the help of the

previous approximation and recalling that $|w_{t,ijq}| \leq 1 - \epsilon'''$ so that $|w_{\epsilon',ijq}| \leq 1$, we see that

$$\begin{aligned} &T \max_i \sum_j \left\| \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left(\mu_{\pm} \left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)} \right) \lambda_{sj} \left[\boldsymbol{x}_{0,j}^{(2l'-2)} - \boldsymbol{x}_{t,j}^{(l'-1)} \right] \right) \right\|_{\infty} \\ &\leq T N_t \max_j \max_q |x_{t,jq}^{(l'-1)} - x_{0,jq}^{(2l'-2)}| \leq \epsilon'/3 \end{aligned} \tag{9}$$

according to our assumption on $\max_{j,q} |x_{t,jq}^{(l'-1)} - x_{0,jq}^{(2l'-2)}|$.

Third, let us bound the activation function approximation error in Eq. (7), which we control by the initialization constant $\sigma > 0$, which we can make arbitrarily small.

$$\begin{aligned} &T \max_i \sum_j \left\| \sum_{s=1}^{c_{1/2}} \boldsymbol{W}_{\epsilon',is}^{(2l')} * \left[\mu_{\pm} \left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)} \right) \lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)} - \phi \left(\lambda_{sj} \boldsymbol{x}_{0,j}^{(2l'-2)} \right) \right] \right\|_{\infty} \\ &\leq T \sum_j \sum_q \sum_s |w_{\epsilon',isq}^{(2l')}| \epsilon'' \leq T N_t C \log \left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}} \right) (r/\sigma) \epsilon'' \end{aligned} \tag{10}$$

We have to choose $\sigma = a(\epsilon'')/2$ small enough so that

$$\epsilon'' \leq \frac{\epsilon'}{3TN_tC \log \left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}} \right) \frac{r}{\sigma}} = \frac{\epsilon'}{3TN_tC \log \left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}} \right) \frac{r}{2a(\epsilon'')}}. \tag{11}$$

Note that we can find an appropriate ϵ'' because the function $g(\epsilon'') = \frac{\epsilon''}{a(\epsilon'')}$ is invertible on a suitable interval $]0, \epsilon']$. We can therefore define

$$\epsilon'' = g^{-1} \left(\frac{\epsilon'}{3TN_tC \log \left(\frac{N_t}{\min\{\epsilon'/(3T), \delta'/2\}} \right) \frac{r}{2}} \right). \tag{12}$$

With this we can conclude that the LT approximates the target network up to error ϵ'. □

Interestingly, note that the activation function approximation does not directly impact our width requirement. It relies, however, on a suitable parameter initialization approach.

($2L_t$)-construction Stacking L_t layers together, we can prove our LT existence theorem for the ($2L_t$)-construction.

Statement (LT existence ($2L_t$)-construction)). Assume that $\epsilon, \delta \in (0, 1)$, a convolutional target network (without skip connections) $f_t(x) : \mathcal{D} \subset \mathbb{R}^{c_0 \times d_0} \to \mathbb{R}^{c_L \times d_{L_t}}$ with architecture $\bar{c}_t$ of depth L_t with $N_{t,l}$ nonzero parameters in Layer l, and a source network f_0 with architecture $\bar{n}_0$ of depth $L_0 = 2L_t$ are given. Let ϕ be the activation function of f_t with Lipschitz constant T fulfilling Assumption 2.3 with $d = 0$. Then, with probability at least $1 - \delta$, f_0 contains a subnetwork $f_\epsilon \subset f_0$ so that each output component i is approximated as $\max_{\boldsymbol{x} \in \mathcal{D}} |f_{t,iq}(\boldsymbol{x}) - f_{\epsilon',iq}(\boldsymbol{x})| \leq \epsilon$ if for all $l' \in [L_t]$

$$c_{0,2l'+1} \geq C c_{t,l} \log \left(\frac{N_t}{\min\{\epsilon / \prod_{s=l}^{L_t} (3TN_{t,s}), \delta\}} \right),$$

and $n_{0,2l'} \geq n_{t,l'} + 1$, and if the parameters of f_0 are initialized according to Assumption 2.1 with $\sigma_{2l'+1} = r/\sigma_{2l'}$ and $\sigma_{2l'} = a(\epsilon'')/2$ and $\epsilon'' = g^{-1} \left(\frac{\epsilon'}{CN_t \log \left(\frac{N_t}{\min\{\epsilon / \prod_{s=l}^{L_t} (3TN_{t,s}), \delta\}} \right)} \right)$ for $g(\epsilon'') = \epsilon''/(a(\epsilon''))$.

Proof. The proof is a repeated application of Corollary C.1. Only the first layer approximation is special, as we might not have a constant tensor available among the data inputs to approximate the target biases of the first layer. In this case, we need to modify our bias approximation by using the nonzero biases of f_0 in Layer $l = 1$. Instead of pruning univariate

tensors in the first layer that take a constant tensor as input, the neurons that we reserve in Layer 1 of the LT with indices I_b receive completely zero weights but keep a bias term $b_{\epsilon,s} = b_{0,s}$. The remaining steps of the proof for the first layer are identical to the proof of Corollary C.1.

It is only left to show how to adapt the ϵ' and δ' to account for multiple layer approximations. $\delta' = \delta/N_t$ with $N_t = \sum_l N_{t,l}$ is sufficient to ensure that all subset sum approximations of all parameters are successful with probability δ. The adaptation of the error, however, needs to take into account how error propagates through different layers. To approximate the output successfully, Corollary C.1 requires that each input tensor element $x_{t,iq}^{(L-1)}$ can have an error of maximally $\epsilon/(3TN_{t,L})$. Repeating this argument inductively, results in an allowed error of $\epsilon_l = \frac{\epsilon}{(3T)^{L_t-l+1}\prod_{s=l}^{L_t} N_{t,s}}$ in the approximation of the target layer l. □

Note that with stricter assumptions on the target parameters, we could also obtain a more favorable scaling of the error with the number of nonzero parameters.

C.2. Proof of Thm. 3.2 ($L_t + 1$)-construction)

Statement (LT existence ($L_t + 1$)-construction)). Assume that $\epsilon, \delta \in (0, 1)$, a convolutional target network (possibly with skip connections) $f_t(x) : \mathcal{D} \subset \mathbb{R}^{c_0 \times d_0} \to \mathbb{R}^{c_L \times d_{L_t}}$ with architecture $\bar{c}_t$ of depth L_t, and a source network f_0 with architecture $\bar{n}_0$ of depth $L_0 = L_t + 1$ are given. Let ϕ be the activation function of f_t and f_0 with Lipschitz constant T. Let further ϕ_0 be the activation function of f_0 in the first layer fulfilling Assumption 2.3 with $d = 0$. Define the number N_l of effective nonzero parameters in Layer l as $N_l = N_{w,l} + N_{m,l}$. Then, with probability at least $1 - \delta$, f_0 contains a subnetwork $f_\epsilon \subset f_0$ so that each output component is approximated as $\max_{\boldsymbol{x}\in\mathcal{D}} |f_{t,iq}(\boldsymbol{x}) - f_{\epsilon',iq}(\boldsymbol{x})| \leq \epsilon$ if for all $l \in [L_t]$

$$c_{0,l+1} \geq C c_{t,l} \log\left(\frac{1}{\min\{\epsilon_l, \rho\delta/N_l\}}\right),$$

and $n_{0,1} \geq C c_{t,0} \log\left(\frac{1}{\min\{\epsilon_1, \delta\rho\}}\right)$ with $\rho = C N_l^{1+\gamma} \log(1/\min\{\min_l \epsilon_l, \delta\})$ for any $\gamma > 0$. $\epsilon_l = \frac{\epsilon}{2TN_{w,l}\prod_{s=l+1}^{L}(2(TN_{w,s}+N_{m,s}))}$. Additionally, we require that the parameters of f_0 are initialized according to Assumption 2.1 with $\sigma_l = 1$ for $l > 2$, $\sigma_1 = r/\sigma_2$ and $\sigma_2 = a(\epsilon'')/2$ and $\epsilon'' = g^{-1}\left(\frac{\epsilon'}{C N_l \log\left(\frac{N_l}{\min\{\epsilon_l,\delta\}}\right)}\right)$ for $g(\epsilon'') = \epsilon''/(a(\epsilon''))$.

Proof. In contrast to Thm. 3.1, in the approximation of the target layers $l > 1$, we do not need to approximate the activation function locally as a Leaky ReLU. Thus, we save the approximation error and the separate approximation of positive and negative parts. Moreover, we can use smaller base sets to solve subset sum approximation problems because the random variables are distributed as $X_k \sim U[-1, 1]$ instead of $X_k \sim U[0, 1]U[-1, 1]$.

Another advantage of this construction is that skip connections can be naturally integrated. As a consequence, we also have to consider the error that we inflict by pruning or just representing skip connections. Let us regard the error at Layer l and denote with ϵ_l the maximal error of a parameter approximation. Similar to before, we have

$$\begin{aligned}\left\|x_t^{(l)} - x_0^{(l+1)}\right\|_\infty &\leq TN_{w,l}\epsilon_l + TN_{w,l}\left\|x_t^{(l-1)} - x_0^{(l)}\right\|_\infty + N_{m,l}\max_{s\leq l-1}\left\|x_t^{(s)} - x_0^{(s+1)}\right\|_\infty \\ &\leq TN_{w,l}\epsilon_l + (TN_{w,l} + N_{m,l})\max_{s\leq l-1}\left\|x_t^{(s)} - x_0^{(s+1)}\right\|_\infty .\end{aligned} \tag{13}$$

In fact, it also follows that

$$\max_{s\leq l}\left\|x_t^{(s)} - x_0^{(s+1)}\right\|_\infty \leq TN_{w,l}\epsilon_l + (TN_{w,l} + N_{m,l})\max_{s\leq l-1}\left\|x_t^{(s)} - x_0^{(s+1)}\right\|_\infty . \tag{14}$$

The error of the last layer can therefore be bounded by ϵ if we ensure that $\epsilon_L = \epsilon/(2TN_{w,l})$ and $\max_{s\leq L-1}\left\|x_t^{(s)} - x_0^{(s+1)}\right\|_\infty \leq \epsilon/(2(TN_{w,L} + N_{m,L}))$. We can thus derive the error by propagating it from Layer l to $l - 1$. This leads to the definition $\epsilon_l = \frac{\epsilon}{2TN_{w,l}\prod_{s=l+1}^{L}(2(TN_{w,L}+N_{m,L}))}$.

In addition, we need to investigate how many more subset sum approximations we have to solve in this construction. The argument is very similar to the one for fully-connected networks (Burkholz, 2022). For completeness, we repeat it here.

δ is modified by $\rho \geq \rho' = \sum_{l=1}^{L} \rho'_l$, where ρ' counts the increased number of required subset sum approximation problems to approximate the L target layers with our lottery ticket and ρ_l counts the same number just for Layer l.

For each non-zero parameter, we will need two solve at least one subset sum approximation problem or sometimes two in case of the first target layer. We denote the number of non-zero parameters in Layer l as N_l. Thus, if our target network is dense without skip connections and all parameters are nonzero, we have $N_l = c_{t,l}c_{t,l-1}k_{t,l}$ and in total $N_t = \sum_{l=1}^{L} c_{t,l}(c_{t,l-1}k_{t,l} + 1)$.

Let us start with counting the number ρ'_L of required subset sum approximation problems in the last layer because it determines how many neurons we need in the previous layer. This in turn defines how many subset sum approximation problems we have to solve to construct this previous layer.

The last layer requires us to solve exactly $\rho'_L = N_L$ subset sum problems, which can be solved successfully with high probability if $c_{0,L-1} \geq Cc_{t,L-1}\log(1/\min\{\epsilon_L, \delta/\rho'\})$. We will only need to construct a subset of these neurons with the help of Layer $L-2$, i.e., exactly the neurons that are used in the lottery ticket. If $c_{0,L-1}$ is large, this might require only $2-3$ neurons per parameter. For simplicity, however, we bound this number by the total number of available channels. To reconstruct one set of channels, we need approximate N_{L-1} parameters. As we have to maximally construct $C\log(1/\min\{\epsilon_L, \delta/\rho'\})$ sets of these channels, we can bound $\rho'_{L-1} \leq CN_{L-1}\log(1/\min\{\epsilon_L, \delta/\rho'\}$.

Note that we can solve all of these subset sum approximation problems with the help of $c_{t,L-2} \geq CN_{L-2}\log(1/\min\{\epsilon_{L-1}, \delta/\rho'\}$ of neurons and this number does not scale by the fact that we have to construct not only $c_{t,L-1}$ channels but a number that is increased by a logarithmic factor. The higher number of required neuron approximations only affects the number of required subset sum approximation problems and thus the needed success probability of each parameter approximation via ρ.

Repeating the same argument for every layer, we derive $\rho'_l \leq CN_l\log(1/\min\{\epsilon_{l+1}, \delta/\rho'\}$, which could also be shown formally by induction. In total we thus find $\rho' = \sum_{l=1}^{L} \rho'_l \leq CN_l\log(1/\min\{\min_l \epsilon_l, \delta/\rho'\}) \leq CN_t\log(1/\min\{\min_l \epsilon_l, \delta/\rho\})$. A ρ that fulfills $\rho = CN_t\log(1/\min\{\min_l \epsilon_l, \delta/\rho\})$ would therefore be sufficient to prove our claim. $\rho = CN_t^{1+\gamma}\log(1/\min\{\epsilon, \delta\})$ for any $\gamma > 0$ works, as $CN_t^{\gamma} \geq \log(N_t)$. □

D. Activation functions with $d \neq 0$

As explained for fully-connected networks by (Burkholz, 2022), our derivations also apply to activation functions with $\phi(0) = d \neq 0$ if we initialize the parameters in our source network with the 'looks-linear' initialization (Burkholz & Dubatovka, 2019).

E. Strides

The main objective of pruning the first layer in Corollary C.1 is to create multiple versions of the input tensors. This can be achieved by pruning 1-dimensional filters with stride 1. If the stride is higher and the filter dimensions is big enough so that filter windows overlap, we can always prune the available filter down to an univariate one - with the given stride s. Yet, with such a filter, if the stride is $s > 1$, we will not create a complete version of an input filter. If we multiply our width requirement by the stride s, we can still reconstruct it by adding partial input filter versions. This can be achieved by pruning a filter in the source network at different positions. For each additional position, we need another univariate filter, which explains our increased width requirement.

F. LT Construction Algorithm

To verify the validity of our derivations and theoretical existence results, we have explicitly constructed LTs. The algorithm follows the same steps as our constructive proofs. Different from the theoretical probabilistic statement, however, we do not assess the construction probability but create LTs of variable width that take failed construction attempts of neurons into account. The reason for this design choice is that it can be computational demanding to reconstruct a large target network. Instead of starting a new reconstruction at each failed approximation, we extend the width of the target network to create the possibility to compensate for failed constructions.

Given inputs are: a target network f_t, a randomly initialized source network f_0, a subset sum base set size m, and an allowed approximation error ϵ that we can make in the construction of each target parameter. Note that, for convenience, $\epsilon > 0$ is

here defined as an error bound of the approximation of a single target parameter and not the global error of the network approximation. Based on these inputs, we derive a LT f_ϵ as subnetwork of the random source network f_0, which can be defined by a mask $M_{l,\epsilon}$ that is applied element-wise to the parameters of f_0, as described in Algorithms 1 and 2. We start in the layers closest to the input and approximate each parameter by solving an independent subset sum approximation problem (SSAP).

A Pytorch implementation is available in the Github repository RelationalML/LT-existence.

Computational complexity: The construction algorithm is linear in the number of target parameters. However, the constant slope depends exponentially on the size of the base set of each subset sum approximation problem and thus the degree of overparametrization of the mother network relative to the target network. Note that the complexity is independent of the input dimension for convolutional layers. The input dimension only influences the complexity of evaluating a lottery ticket (LT) on data.

Algorithm 1 $2L$ approximation

Input: target f_t of depth L, source f_0 of depth $2L$ width $c_{0,2l} = c_{t,l} + m$, $c_{0,2l-1} = (2m+1)c_{t,l}$ with base set size m, allowed error per parameter approx. $\epsilon > 0$.
for $l = 1$ **to** L **do**
  for $i = 1$ **to** $c_{t,l}$ **do**
    Approximate every filter weight leading to channel i:
    for $j = 1$ **to** $c_{t,l-1}$ **do**
      Find index permutation $I(j')$ of channels in Layer $2l-1$ of f_0 so that $w^{(2l-1)}_{0,I((j'-1)2m+s)1} > 0$, $w^{(2l-1)}_{0,I((2j-1)m+s)1} < 0$, and $w^{(2l-1)}_{0,I(2mc_{l-1}+s)j1} > 0$ for all $s \in [m] = \{1, ..., m\}$ if $l > 1$ or $b^{(1)}_{0,I(2mc_{l-1}+s)} > 0$ if $l = 1$. If not possible reduce m of corresponding block.
      for $k = 1$ **to** $k_{t,l}$ **do**
        Solve SSAP for the positive part:
        Define $x_s = w^{(2l-1)}_{0,I((j-1)2m+s)1} w^{(2l)}_{0,iI((j-1)2m+s)k}$.
        Define $P = \{S \subset [m] \mid |\sum_{s\in S} x_s - w^{(l)}_{t,ijk}| \leq \epsilon\}$.
        Solve $S^*_+ = \text{argmin}_{S\in P}|S|$.
        Solve SSAP for the negative part:
        Define $x_s = w^{(2l-1)}_{0,((2j-1)m+s)1} w^{(2l)}_{0,iI((2j-1)m+s)k}$.
        Define $P = \{S \subset [m] \mid |\sum_{s\in S} x_s - w^{(l)}_{t,ijk}| \leq \epsilon\}$.
        Solve $S^*_- = \text{argmin}_{S\in P}|S|$.
        Solve SSAP for the bias:
        if $l > 1$ **then**
          Define $x_s = w^{(2l-1)}_{0,I(2mc_{l-1}+s)j1} w^{(2l)}_{0,iI(2mc_{l-1}+s)k}$.
        else
          Define $x_s = b^{(1)}_{0,I(2mc_{l-1}+s)} w^{(2l)}_{0,iI(2mc_{l-1}+s)k}$.
        end if
        Define $P = \{S \subset [m] \mid |\sum_{s\in S} x_s - w^{(l)}_{t,ijk'}| \leq \epsilon\}$.
        Solve $S^*_b = \text{argmin}_{S\in P}|S|$.
      end for
    end for
    if All approximations involving i are successful: **then**
      Define LT mask:
      Initialize the masks $M^{(2l-1)} \in 0^{c_{0,2l} \times c_{0,2(l-1)} \times k_{t,l}}$.
      Set $m^{(2l-1)}_{0,I((j'-1)2m+s)1} = 1$ for $s \in S^*_+$, $m^{(2l-1)}_{0,I((2j-1)m+s)1} = 1$ for $s \in S^*_-$, and $m^{(2l-1)}_{0,I(2mc_{l-1}+s)j1} = 1$ for $s \in S^*_b$ if $l > 1$ or the bias mask accordingly.
      Set $m^{(2l)}_{0,iI((j-1)2m+s)k} = 1$ for $s \in S^*_+$, $m^{(2l)}_{0,iI((2j-1)m+s)k} = 1$ for $s \in S^*_-$, and $m^{(2l)}_{0,iI(2mc_{l-1}+s)k} = 1$ for $s \in S^*_b$.
    else
      Repeat approximation attempt of channel i with next channel i' in f_0 until successful. Concatenate width of f_0 accordingly (and re-index for convenience so that target node i is approximated by source node i).
    end if
  end for
  if $l < L$ **then**
    Create m constant neurons for bias construction in next layer as above with $b_{t,l} = 1$.
  end if
end for
Output: masks $M^{(l)}$, final width of f_0.

Algorithm 2 $L+1$ approximation

Input: target f_t of depth L, source f_0 of depth $L+1$ and width $c_{0,1} = (2m+1)c_0$, $c_{0,l+1} = m(c_{t,l}+1)$ with base set size m for $l < L$, allowed error per parameter approx. $\epsilon > 0$.
Approximate target Layer $l = 1$ **by two masked source layers with Algorithm 1.**
for $l = 2$ **to** L **do**
  for $i = 1$ **to** $c_{t,l}$ and constant channel ($b_{t,c_{t,l}+1} = 1$) **do**
    Create m **approximate copies of each target channel** i**:**
    for $q = 1$ **to** m **do**
      Approximate every filter weight leading to channel i:
      for $j = 1$ **to** $c_{t,l-1}$ **do**
        for $k = 1$ **to** $k_{t,l}$ **do**
          Solve SSAP:
          Define $x_s = w^{(l+1)}_{0,((i-1)m+q)((j-1)m+s)k}$ for $s \in [m]$.
          Define $P = \{S \subset [m] \mid |\sum_{s\in S} x_s - w^{(l)}_{t,ijk}| \leq \epsilon\}$ or $P = \{S \subset [m] \mid |\sum_{s\in S} x_s - 1| \leq \epsilon\}$ for constant channels.
          Solve $S^* = \text{argmin}_{S\in P}|S|$.
        end for
      end for
      if All approximations involving copy q of i are successful: **then**
        Define LT mask:
        Initialize the masks $M^{(l+1)} \in 0^{c_{0,l+1}\times c_{0,l}\times k_{t,l}}$.
        Set $m^{(l+1)}_{0,((i-1)m+q)((j-1)m+s)k} = 1$ for $s \in S^*$ or the bias mask accordingly.
      else
        Repeat approximation attempt of copy q of channel i with next channel i' in f_0 until successful. Concatenate width of f_0 accordingly (and re-index for convenience so that target node i is approximated by source node i).
      end if
    end for
  end for
end for
Output: masks $M^{(l)}$, final width of f_0.