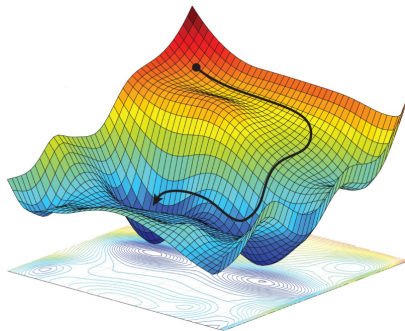


Особенности настройки глубоких нейросетей

Виктор Китов

victorkitov.github.io



Содержание

- 1 **Борьба с переобучением**
- 2 Численные методы оптимизации 1го порядка
- 3 Перенормировка активаций (batch normalization)
- 4 Ограничение градиента
- 5 Дополнительная тема

Расширение обучающей выборки

- Модели глубокого обучения имеют много параметров.
 - полносвязный слой $A \rightarrow B$ нейронов содержит $(A + 1) B$ связей!
- Методы борьбы с переобучением:
 - предобучить начальные слои по большим доступным выборкам схожей задачи
 - например, по ImageNet (>14 миллионов размеченных объектов) для изображений
 - domain adaptation: адаптация модели одной доменной области к другой
 - использовать расширение (augmentation) обучающей выборки: $(x, y) \rightarrow \{(\phi_1(x), y), (\phi_2(x), y), (\phi_3(x), y), \dots\}$
 - $\phi_1, \phi_2, \phi_3, \dots$ - инвариантные к отклику преобразования
 - например, изменения яркости/контрастности, поворот, обрезка.

Регуляризация

Бороться с переобучением можно упрощая модель:

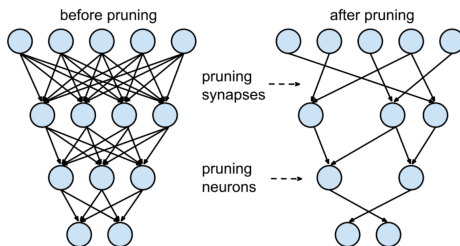
- использовать меньше нейронов/связей
- использовать регуляризацию
 - L_1, L_2 (можно со своим весом на каждом слое)
- ограничения на веса при настройке
 - $w_i = w_j$ либо регуляризация $+\lambda|w_i - w_j|^2$
- DropOut, DropConnect
- ранняя остановка (early stopping)
- обучение с зашумлёнными входами
- дистилляция знаний (2 сети: учитель, студент)

Уменьшение вычислительной сложности

Для уменьшения вычислительной сложности:

- можно обучить большую сеть, а потом ее прореживать:
 - отбросить связи с низким весом
 - отбросить связи, слабо влияющие на ф-цию потерь (optimal brain damage)
 - отбросить нейроны с низкими входными весами
 - отбросить нейроны со слабыми активациями

Прореживание связей и нейронов:



Optimal brain damage: связи, слабо влияющие на $L(w)$

$$\begin{aligned} L(w) - L(w^*) &\approx \nabla L(w^*)^T (w - w^*) \\ &+ \frac{1}{2} (w - w^*)^T \nabla^2 L(w^*) (w - w^*) \end{aligned}$$

Optimal brain damage: связи, слабо влияющие на $L(w)$

$$\begin{aligned}
 L(w) - L(w^*) &\approx \nabla L(w^*)^T (w - w^*) \\
 &\quad + \frac{1}{2} (w - w^*)^T \nabla^2 L(w^*) (w - w^*) \\
 &\approx \frac{1}{2} (w - w^*)^T \nabla^2 L(w^*) (w - w^*) \approx \frac{1}{2} \sum_i \frac{\partial L(w^*)}{\partial w_i^2} (w_i - w_i^*)^2
 \end{aligned}$$

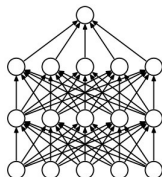
Приближения:

- $O(\|w - w^*\|^3) \approx 0$
- в оптимуме $\nabla L(w^*) = 0$
- $\nabla^2 L(w^*)$ диагональна-игнорируем смешанные производные

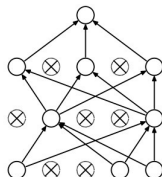
Алгоритм optimal brain damage:

- повторять нужно число раз
 - обучить нейросеть
 - отбросить связи с $\frac{\partial L(w^*)}{\partial w_i^2} (w_i - w_i^*)^2 < threshold$

DropOut: обучение



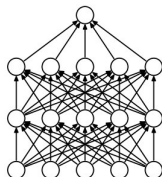
(a) Standard Neural Net



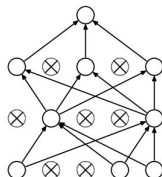
(b) After applying dropout.

- Для каждого минибатча каждый нейрон независимо, кроме выходных, отбрасывается с вероятностью $(1 - p)$ и оставляется с вероятностью p .

DropOut: обучение



(a) Standard Neural Net



(b) After applying dropout.

- Для каждого минибатча каждый нейрон независимо, кроме выходных, отбрасывается с вероятностью $(1 - p)$ и оставляется с вероятностью p .
 - ускоряет обучение;
 - уменьшает переобучение, препятствуя со-настройке нейронов;
 - по сути, учим ансамбль прореженных моделей.

DropOut: применение

- При обучном обучении нейрон получает на вход:

$$\sum_i w_i a_i$$

DropOut: применение

- При обучном обучении нейрон получает на вход:

$$\sum_i w_i a_i$$

- При обучении с DropOut:

$$\sum_i (p w_i \cdot a_i + (1 - p) w_i \cdot 0) = \sum_i p w_i a_i$$

DropOut: применение

- При обучном обучении нейрон получает на вход:

$$\sum_i w_i a_i$$

- При обучении с DropOut:

$$\sum_i (p w_i \cdot a_i + (1 - p) w_i \cdot 0) = \sum_i p w_i a_i$$

- Применение модели: оставляются все нейроны, но выход каждого нейрона домножается на p .
 - для уменьшения вычислений можно:
 - обучение: выход a_i/p , когда a_i оставлен.
 - прогноз: выход a_i без изменений.
- Рекомендация: более ранние слои слабее прореживать
 - прореживание 0-го слоя: м-д случайных подпространств

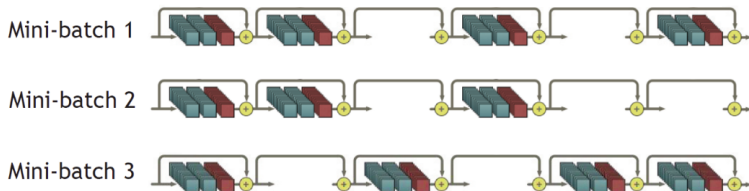
Стохастическая глубина в сетях¹

Deep Networks with Stochastic Depth - используется случайное отбрасывание слоёв в классификации изображений.

Полная архитектура (используется на тесте):



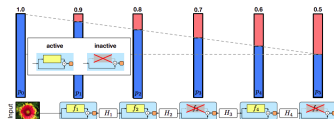
Обучение: отбрасывается преобразования блока l с вероятностью $1 - p_l$



¹<https://arxiv.org/abs/1603.09382>

Stochastic Depth

Рекомендуемая зависимость вероятности оставления блока p_l от слоя l :



- Мотивация p_l : более ранние слои извлекают более базовые признаки, используемые всеми последующими слоями.
- Применение: используются сеть со всеми блоками, домноженными на p_l .
- Мотивация архитектуры:
 - быстрее обучение (менее глубокая сеть на каждом минибатче)
 - борьба с переобучением (когда всё зависит от взаимосвязей между отдельными блоками)
 - по сути обучается ансамбль моделей разной глубины.

Содержание

- 1 Борьба с переобучением
- 2 Численные методы оптимизации 1го порядка
- 3 Перенормировка активаций (batch normalization)
- 4 Ограничение градиента
- 5 Дополнительная тема

Особенности оптимизации нейросетей

- Зависимость $\hat{y}(x)$ в общем случае невыпукла.
- $\mathcal{L}(\hat{y}, y)$ - невыпукла \Rightarrow много локальных минимумов.
- На найденный минимум влияют:
 - начальное приближение
 - объекты минибатчей
 - метод обучения и динамика ε_t
- Можно настраивать разными способами, а потом
 - выбрать наилучшее решение по валидации
 - усреднить несколько решений (ансамбль)

Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

- медленно, не применим для динамических данных
- Stochastic gradient descent (SGD): стохастический спуск с сэмплированием по 1 объекту

$$w_{t+1} := w_t - \eta \nabla L_w(w; x_i, y_i)$$

- берём последовательные объекты, но перемешиваем выборку перед каждой эпохой.

Базовые градиентные методы

- Batch gradient descent: градиентный спуск по всем объектам выборки

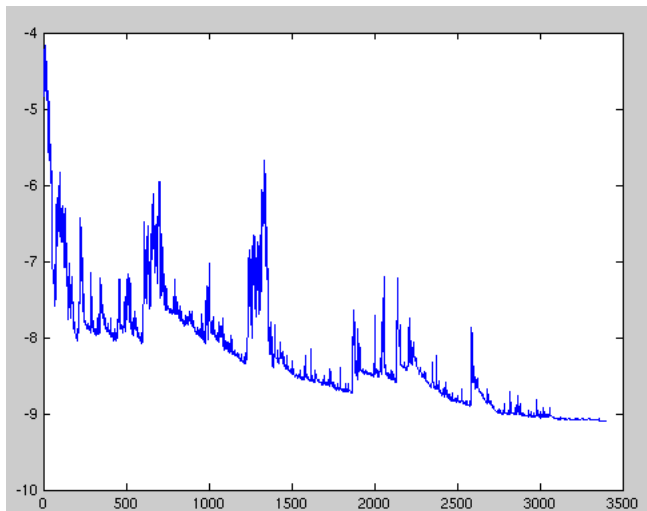
$$w_{t+1} := w_t - \eta \nabla_w L(w; X, Y)$$

- медленно, не применим для динамических данных
- Stochastic gradient descent (SGD): стохастический спуск с сэмплированием по 1 объекту

$$w_{t+1} := w_t - \eta \nabla L_w(w; x_i, y_i)$$

- берём последовательные объекты, но перемешиваем выборку перед каждой эпохой.
- SGD быстрее сходится, но
 - неустойчивая оценка градиента
 - слабое использование параллелизации

Пример сходимости SGD



Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

- точнее оценки градиента
- и быстрее: параллелизация вычислений по минибатчу

Базовые градиентные методы

- Minibatch gradient descent: стохастический спуск с сэмплированием по набору объектов

$$w_{t+1} := w_t - \eta \nabla_w L(w; x_{i+1:i+K}, y_{i+1:i+K})$$

- точнее оценки градиента
- и быстрее: параллелизация вычислений по минибатчу
- Сложности:
 - нужен выбор динамики убывания η
 - одинаковый шаг для разных весов
 - логичнее веса брать меньше, где ф-ция резко меняется и отвечающие редко встречающимся признакам.
 - застревание в локальных оптимумах и точках перегиба

Модификации инерции и Нестерова

- SGD с инерцией (momentum), "мяч катится с горы":

$$v_t := \gamma v_{t-1} + \eta \nabla_w L(w_t)$$

$$w_{t+1} := w_t - v_t$$

Модификации инерции и Нестерова

- SGD с инерцией (momentum), "мяч катится с горы":

$$v_t := \gamma v_{t-1} + \eta \nabla_w L(w_t)$$

$$w_{t+1} := w_t - v_t$$

- устойчивее градиент (усредняем по градиентам с прошлых итераций)
- можем брать выше шаг обучения, ускорение

Модификации SGD

- Обозначим $g_t = \nabla_w L(w_t)$; $w, g_t \in \mathbb{R}^K$. Операции над векторами поэлементные.
- AdaGrad ($\varepsilon = 10^{-6}$)

$$G_t := G_t + \nabla_w L(w_t)^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot \nabla_w L(w_t)$$

- RMSprop

$$G_t := \gamma G_{t-1} + (1 - \gamma) \nabla_w L(w_t)^2$$
$$w_{t+1} := w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot \nabla_w L(w_t)$$

Модификации SGD

- Adam=RMSprop+инерция
($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$):

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) \nabla_w L(w_t)$$

$$G_t := \beta_2 G_{t-1} + (1 - \beta_2) \nabla_w L(w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{G}_t = \frac{G_t}{1 - \beta_2^t}$$

$$w_{t+1} := w_t - \frac{\eta}{\sqrt{\hat{G}_t} + \varepsilon} \cdot \hat{m}_t$$

- Nadam: Adam+Nesterov Accelerated Gradient.

Модификации SGD

- AMSGrad: позволяет помнить историю без экспоненциального забывания.
 - при этом перенормировка m_t, v_t не используется.

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) \nabla_w L(w_t)$$

$$G_t := \beta_2 G_{t-1} + (1 - \beta_2) \nabla_w L(w_t)^2$$

$$\hat{G}_t = \max(\hat{G}_{t-1}, G_t)$$

$$w_{t+1} := w_t - \frac{\eta}{\sqrt{\hat{G}_t + \varepsilon}} \odot \hat{m}_t$$

Дополнительные улучшения²

- Ранняя остановка (early stopping) - борьба с переобучением
- Добавление шума к градиенту - находим оптимум с большей окрестностью:

$$g_t := g_t + \mathcal{N}(0, \sigma_t^2)$$

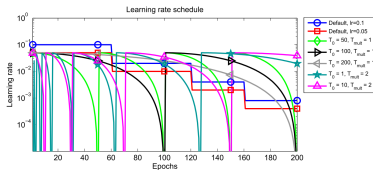
$$\sigma_t^2 = \frac{\eta}{(1+t)^\gamma}$$

- Обучение по расписанию (curriculum learning)
 - сначала обучаемся на простых объектах, потом на сложных
- Параллелизация вычислений SGD.
- Ускорение обучения: batch normalization.

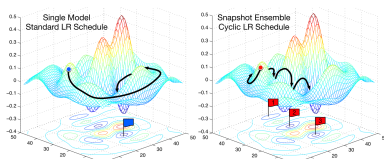
²Больше информации: <https://ruder.io/deep-learning-optimization-2017/>

Дополнительные улучшения

- LR schedule - закон изменения η . Подобный закон, приводит к к оптимуму с большей окрестностью:



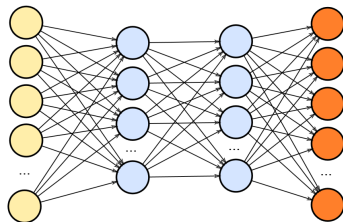
- Перед каждым увеличением η получили некоторую модель \Rightarrow можем модели комбинировать в композицию:



Содержание

- 1 Борьба с переобучением
- 2 Численные методы оптимизации 1го порядка
- 3 Перенормировка активаций (batch normalization)**
- 4 Ограничение градиента
- 5 Дополнительная тема

Перенормировка активаций: мотивация



- SGD $w := w - \varepsilon \nabla_w \mathcal{L}(x, y)$ обновляет все веса на всех слоях одновременно.
- Распределение выходов меняется, и поздние слои должны обучаться снова.
- Также вход может сдвинуться в область малых градиентов нелинейности.
- Перенормировка активаций (batch normalization) частично это решает.

BatchNorm: идея

- Нормализуем выходы на промежуточных слоях:

$$\tilde{x}_k = \frac{x_k - \mu_k}{\sigma_k}, \quad \mu_k = \mathbb{E}x_k, \sigma_k = \sqrt{\text{Var}(x_k)}$$

- гарантируем $\mathbb{E}\tilde{x}_k = 0$, $\text{Var} \tilde{x}_k = 1$ после обновления весов на предыдущих слоях.
 - обучение быстрее для поздних слоёв

BatchNorm: идея

- Нормализуем выходы на промежуточных слоях:

$$\tilde{x}_k = \frac{x_k - \mu_k}{\sigma_k}, \quad \mu_k = \mathbb{E}x_k, \sigma_k = \sqrt{\text{Var}(x_k)}$$

- гарантируем $\mathbb{E}\tilde{x}_k = 0$, $\text{Var}\tilde{x}_k = 1$ после обновления весов на предыдущих слоях.
 - обучение быстрее для поздних слоёв
- **Обучение:**
 - проблема: не знаем μ_k, σ_k
 - изменяются динамически с обновлением весов
 - решение: оценим по текущему минибатчу (должен быть достаточного размера)
- **Применение:**
 - распределение x_k фиксировано, так что можем оценить μ_k, σ_k по всей обучающей выборке.
 - более эффективно: оценки μ_k, σ_k с последовательности последних минибатчей.

BatchNorm: основной алгоритм

$$\tilde{x}_k = \alpha_k \frac{x_k - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + \beta_k, \quad \mu_k = \bar{x}_k, \quad \sigma_k = \sqrt{Var(x_k)}$$

BatchNorm: основной алгоритм

$$\tilde{x}_k = \alpha_k \frac{x_k - \mu_k}{\sqrt{\sigma_k^2 + \varepsilon}} + \beta_k, \quad \mu_k = \bar{x}_k, \quad \sigma_k = \sqrt{Var(x_k)}$$

- **Обучение:**

- μ_k, σ_k по минибатчу
- α_k, β_k выходное std. отклонение и среднее.
 - обучаются в процессе настройки сети
- мотивация:
 - можем отменить нормализацию (например в задаче предсказания времени суток по фото)
 - возможность лучше подстроиться под нелинейность (не обнулять вход в половине случаев для ReLU)

- **Применение:**

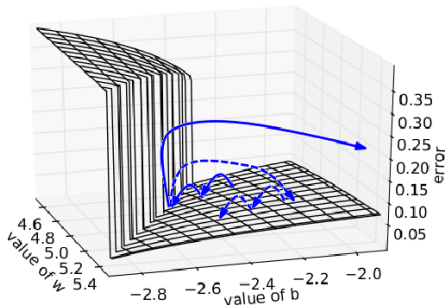
- μ_k, σ_k оценены по широкому классу объектов.
- α_k, β_k фиксированы.

Содержание

- 1 Борьба с переобучением
- 2 Численные методы оптимизации 1го порядка
- 3 Перенормировка активаций (batch normalization)
- 4 Ограничение градиента**
- 5 Дополнительная тема

Проблема большого градиента

Нестабильный по величине градиент мешает сходимости.



Решения - ограничение градиента (gradient clipping) или адаптивное ограничение градиента (adaptive gradient clipping) во время обучения.

Ограничение градиента

Ограничение градиента по порогу (gradient clipping³): гарантия $|\Delta\theta| \leq \varepsilon\lambda$ (помогает в настройке рекуррентных нейросетей)

$$w \rightarrow \begin{cases} w - \varepsilon \nabla_w \mathcal{L}(\hat{y}_i, y_i), & \text{если } \|\nabla_w \mathcal{L}(\hat{y}_i, y_i)\| < \lambda \\ w - \varepsilon \lambda \frac{\nabla_w \mathcal{L}(\hat{y}_i, y_i)}{\|\nabla_w \mathcal{L}(\hat{y}_i, y_i)\|} & \text{если } \|\nabla_w \mathcal{L}(\hat{y}_i, y_i)\| \geq \lambda \end{cases}$$

³<https://arxiv.org/pdf/1211.5063.pdf>

Адаптивное ограничение градиента

Адаптивное ограничение градиента по порогу (adaptive gradient clipping⁴): гарантия $\|\Delta\theta_i\| \leq \varepsilon\lambda \|\theta_i\|$

- помогло настроить SOTA модель классификации ImageNet без батч-нормализации
- Пусть w_i - отдельный параметр (др. вариант применения: w_i - все пар-ры i -го слоя, тогда $\|\cdot\|$ -норма Фробениуса, $\|w_i\|^* = \max(\|w_i\|, \nu)$, $\nu > 0$ - мало).

Для каждого i :

$$w_i \rightarrow \begin{cases} w_i - \varepsilon \nabla_{w_i} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i), & \text{если } \|\nabla_{w_i} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)\| / \|w_i\|^* < \lambda \\ w_i - \varepsilon \lambda \|w_i\|^* \frac{\nabla_{w_i} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)}{\|\nabla_{w_i} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)\|} & \text{если } \|\nabla_{w_i} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)\| / \|w_i\|^* \geq \lambda \end{cases}$$

⁴<https://arxiv.org/pdf/2102.06171.pdf>

Заключение

- Глубокие сети способны сами настраивать сложные признаки.
 - работают лучше, но нужны большие обучающие выборки
- Борьба с переобучением нейросетей:
 - расширение выборки (pretraining, data augmentation)
 - сокращение #нейронов/связей
 - ранняя остановка, L_1/L_2 регуляризация, DropOut.
- Функция потерь невыпукла, возможны локальные оптимумы.
- Идеи улучшений SGD: инерция, ускоренный градиент Нестерова, индивидуальные настраиваемые веса для каждого параметра.
- BatchNorm ускоряет и упрощает сходимость.

Содержание

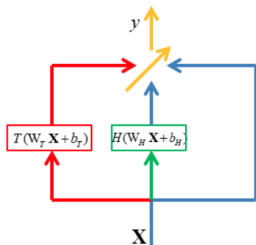
- 1 Борьба с переобучением
- 2 Численные методы оптимизации 1го порядка
- 3 Перенормировка активаций (batch normalization)
- 4 Ограничение градиента
- 5 **Дополнительная тема**

Highway networks⁵ - идея из LSTM

- Для эффективной настройки многослойных сетей в highway networks предлагается использовать блоки:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W}_H) \cdot T(\mathbf{x}, \mathbf{W}_T) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W}_T))$$

- Вариант transform gate: $T(\mathbf{x}, \mathbf{W}_T) = \sigma(\mathbf{W}_T \mathbf{x} + \mathbf{b}_T)$.
 - обобщение: $(1 - T(\mathbf{x}, \mathbf{W}_T)) \rightarrow \text{carry gate } C(\mathbf{x}, \mathbf{W}_C)$.



⁵<https://arxiv.org/pdf/1505.00387.pdf>

Highway networks

- Инициализация: $\mathbf{b}_T = -1, -2, -3$, чтобы на начальных итерациях $\mathbf{y} \approx \mathbf{x}$
 - слой $H(\mathbf{x}, \mathbf{W}_H)$ вводится постепенно, только если он действительно нужен
- Highway networks: быстрее настройка, лучше найденный оптимум.

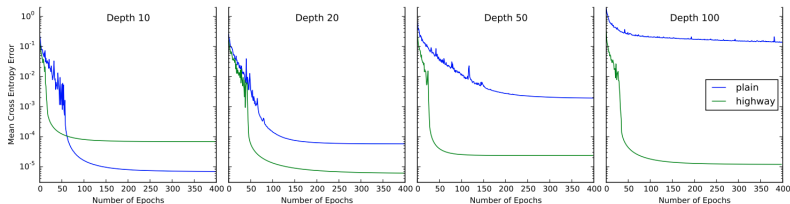


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.